UNIVERSITÉ DU LUXEMBOURG

PhD-FSTC-2016-02
The Faculty of Science, Technology and Communication

Dissertation

Defense held on the 11th January 2016 in Luxembourg

to obtain the degree of

Docteur de l'Université du Luxembourg en Informatique

by

Assaad Moawad
Born on 11th March 1988 in Bayet El Chaar (Lebanon)

Towards Ambient Intelligent Applications Using Models@run.time And Machine Learning For Context-Awareness

Dissertation Defense Committee

Prof. Nicolas Navet, chairman
Professor, University of Luxembourg, Luxembourg

Dr. Francois Fouquet, co-chairman
Research Associate, University of Luxembourg, Luxembourg

Prof. Yves Le Traon, supervisor
Professor, University of Luxembourg, Luxembourg

Prof. Romain Rouvoy, member
Professor, University Lille 1, France

Prof. Houari Sahraoui, member
Professor, University of Montreal, Montreal, Canada

Dr. Patrice Caire, expert
Research Associate, University of Luxembourg, Luxembourg

Abstract

Ambient Intelligence (AmI) constitutes a new paradigm of interaction among humans, smart objects and devices. AmI systems are expected to support humans in their every day tasks and activities. In order to achieve this goal, these systems require augmenting the environment with sensing, computing, communicating, and reasoning capabilities. Due to advances in technology, sensors are getting more powerful, cheaper and smaller, which stimulated large scale development and production. These sensors will generate a big amount of data and can easily lead to millions of values in a short amount of time, which can quickly reach the computation and storage limits when it comes to structuring and processing the data. For this problem, we propose a concept of continuous models that can handle highly-volatile data, and represent the **continuous** nature of sensor data in an efficient and compact way. We show on various AmI datasets that this can significantly improve storage and efficiency.

One important goal of AmI systems is to transform living and working environments into *intelligent spaces* able to adapt to their users' needs and desires in real-time. In this sense, we call AmI applications **context-aware**, meaning that they use environmental information to adaptively provide more relevant and better services to the user. However, AmI systems are composed from heterogeneous components, operating in an open and dynamic environment. Each of these components can have different storage and computation capabilities. They might not have all the information needed to derive context information, and they might not be reachable all the time for various reasons. In this thesis, we present a contextual reasoning solution adapted for component based platforms. Our solution can derive contextual information in a distributed way and can handle inconsistencies when contradictory information is received from several components.

Other than the storage and computation efficiency, several qualities need to be satisfied according to the different contexts. **Privacy** is one of these qualities. AmI services will rely more and more on personal data that is vastly collected, stored, and exchanged with other third parties in order to provide added-value services. Such data are sensitive and often related to personal activities and therefore can lead to privacy risks, especially when data is shared with high precision and frequency. However, this privacy quality can be relaxed in some contexts, for example in an emergency situation in order to increase utility or efficiency. This leads to the need of developing an **adaptive** solution that is able to react to context changes in real-time and involve optimizing conflicting objectives. For this challenge, we propose to use **blurring components** as our main privacy preservation elements. The idea behind this approach is that, by gradually decreasing the data quality, a blurring component is able to hide some of the personal data delivered by sensors while still keeping the necessary information for the services to work. In order to find a good trade-off between these different conflicting objectives, we adapt a multi-objective evolutionary algorithm to run directly on top of domain specific models. We then apply it as our main optimization

engine on models@run.time to keep adapting the different qualities, when the context change.

Finally, AmI services are expected to be tailored for different users' needs in a seamless and unobtrusive way. Meaning that they should be able to detect contexts and learn habits automatically with the least possible intervention of users. In order to achieve this, *machine learning* (ML) techniques need to be merged at the core of reasoning models. These techniques offer powerful tools to automatically detect patterns, categorize contexts, build usage profiles, represent data with compact mathematical hypothesis and provide statistical information vital for the intelligent aspect of AmI. This dissertation ends up by opening new directions on how to model and adapt machine learning techniques to fit for AmI platforms.

Overall, this thesis provides solutions for the next leap of technology, where sensors become ubiquitous in order to empower smart systems. Our solutions, implemented in an open source framework KMF, allow to create efficient and distributed, data and component models for IoT, adaptable at runtime leveraging multi-objective optimization to find a good tradeoff between qualities for the current context, and machine learning techniques to derive contextual rules, profile and learn habits automatically.

Keywords: Distributed, Context-aware, MOEA, Machine Learning, Ambient Intelligence, Internet of Things, Models@run.time, Privacy, Blurring.

Acknowledgements

The PhD experience goes beyond research, experimentations and papers writing. It is indeed a challenging life experience. It is the result of a long process that began mid-January 2012 and which outcome owes much to the support and help of several people. It is a pleasure for me to express my gratitude to them.

First of all, the accomplishments of this challenging experience would have never been possible without the support of my supervisor Prof. Yves Le Traon. I would like to express my deepest gratitude for never stopping in believing in me and encouraging me, especially in the hardest moments of my thesis. He helped me finding new solutions when I only saw closed roads.

Of all the people involved in my thesis, I am particularly grateful to my daily supervisor, Dr. François Fouquet, for his patience, advice and flawless guidance. He showed me how to perform research, and taught me how to conduct rigorous experiments. Most importantly, he helped me to improve my programming skills, and thought me how to work and collaborate in an efficient way with the open source community. I am very happy as well about the friendship we have built up during these years.

My special thanks goes also to the members of my dissertation committee: Prof. Romain Rouvoy and Prof. Houari Sahraoui for investing time to review my work and for providing interesting and valuable feedback. I am equally thankful to my co-supervisor Dr. Jacques Klein. I would like to express my warm thanks as well to all the group members of the SerVal team for the plenty discussions we had. More specifically, the people who helped me during my PhD: Dr. Gregory Nain for the advice he gave me while working with Kevoree and KMF, Thomas Hartmann and Dr. Tejeddine Mouelhi for their continuous feedbacks and reviews. I would also like to extend my thanks to my external co-authors, especially Dr. Nicola Zannone, Vasilis Efthymiou and Dr. Antonis Bikakis.

My PhD experience was a great enriching cultural experience. I enjoyed my stay in the Grand Duchy of Luxembourg, a great place to live, to work and to meet international people. I am thankful to Luxembourgish institutions and particularly to the University of Luxembourg and the city of Luxembourg for all the facilities that made my stay pleasant and joyful. Finally, and more personally, I would like to express my deepest thanks to my family and my friends for their support, especially my photography friends in Luxembourg (Johann Heckel, Bogdan and Mioara) who helped me overcoming the work stress, and my good neighbors in Noertzange (Dimitrios Kampas and Amir Houshang Mahmoudi) for their continuous support.

Assaad Moawad

Luxembourg, Luxembourg, January 2016

Contents

List of abbreviations

AAL	Ambient Assisted Living
AmI	Ambient Intelligence
BLAS	Basic Linear Algebra Subprograms
CDL	Contextual Defeasible Logic
DSML	Domain Specific Modeling Languages
EA	Evolutionary Algorithm
EMF	Eclipse Modeling Framework
IoT	Internet of Things
KMF	Kevoree Modeling Framework
LAPACK	Linear Algebra PACKage
MDE	Model Driven Engineering
ML	Machine Learning
MOEA	Multi-Objective Evolutionary Algorithm
MOO	Multi-Objective Optimization
PbD	Privacy by Design
PET	Privacy Enhancing Technologies

List of figures

List of tables

List of algorithms

1

Introduction

In this chapter, we present the different application domains of this dissertation: internet of things, ambient intelligence, and ambient assisted living. Then we present a generic overview of the common challenges in these domains, as well the different contributions we made to address these challenges throughout the thesis.

Contents

1.1 Application Domains

Due to advances in technology, sensors are getting more powerful, cheaper and smaller in size, which has stimulated large scale development and their deployment in various domains. Smart devices like smart phones are already in our everyday lives and new applications using them are being continuously created. In the near future more and more smart objects will be connected to the Internet. These objects will be equipped with computational capabilities, allowing them to connect and interact with other objects, users or services and adapt their behavior accordingly. This thesis focuses on developing solutions adapted for this leap of technology. First, we define the three main domains that are related to this dissertation: Internet of things, ambient intelligence, and ambient assisted living. Then, we present the common challenges and the contributions made to address these challenges.

1.1.1 Internet of Things

The **Internet of Things** (IoT) paradigm refers to the networked interconnection of everyday objects, which are often equipped with ubiquitous intelligence [AIM10]. IoT will lead to a highly distributed network of communicating devices. Thanks to rapid advances in underlying technologies, IoT is opening tremendous opportunities for a large number of novel applications that promise to improve the quality of our lives [AIM10]. IoT is associated with the following concepts in information and communication technology:

- **Ubiquitous communication** [Wei93]: the general ability of objects to communicate (anywhere and anytime), it is the process by which communications between multiple agents can happen simultaneously and without the restrictions of time.
- **Pervasive computing**: the enhancement of objects with processing power (the environment around us becomes able to compute). In 1991, Mark Weiser, described a vision for 21st century computing that countered the ubiquity of personal computers. "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it." [Wei91].
- **Intelligence**: the capability of objects to register changes in the physical environment and thus actively interact in the process.

Typically, objects fulfilling these requirements are called "smart objects". Moreover, market research has shown a significant growth of sensor deployments over the past decade and has predicted a significant increment of the growth rate in the future [AIM10].

1.1.2 Ambient Intelligence

Ambient Intelligence (AmI) constitutes a new paradigm of interaction among humans, smart objects and devices. AmI systems foster a human–machine model of interaction. AmI systems are expected to support humans in their every day tasks and activities in

a personalized, user-centric, adaptive, seamless and unobtrusive fashion. Its goal is to transform living and working environments into *intelligent spaces* able to adapt to changes in contexts and to their users' needs and desires in near real-time. In order to achieve this goal, these systems require augmenting the environment with sensing, computing, communicating, and reasoning capabilities. In AmI, technologies are deployed to make computers disappear in the background, while the human user moves into the foreground in complete control of the augmented environment [RF05].

The conceptual difference between AmI and IoT is the focus [RF05]. IoT is focused on the technological infrastructure for making things interactive, identifiable and connected. While AmI is a technology focused on the invisible interaction with humans. However, IoT is one of the options for realizing AmI [RF05]. In a "smart house", the field bus systems can be used without a connection to the Internet. But when the smart house is connected, more services can be provided by third parties. In this thesis, we target the IoT for the AmI domain.

1.1.3 Ambient Assisted Living

Ambient Assisted Living (AAL) encompasses technical systems specialized to support elderly people and people with special needs in their daily routine. The main goal of AAL is to maintain and foster the autonomy of those people and, thus, to increase safety in their lifestyle and in their home environment [DMOD+10]. The necessity for such applications arises from the demographic change in industrialized countries where life expectancy is on the rise and the birth rate is in decline [DMOD+10]. These circumstances require innovative and cost-effective solutions to keep the health care expenditures within the bounds of economic possibilities. AAL applications include services, products and concepts to increase the quality of life, wellbeing and safety of elderly people.

The main goal of AAL is to achieve benefits for the individual (increasing safety wellbeing), the economy (higher effectiveness of limited resources) and the society (better living standards) [Geo08]. In this perspective, AAL can be seen as a special case of AmI, where health issues can cause emergency situations and systems need to be very responsive.

1.2 Challenges

In order to develop applications for the AmI and AAL domains based on the IoT paradigm, several challenges need to be addressed. These challenges have been enumerated in many recent surveys, for instance: [CAJ09], [Rou08], [ACRV13]. In this section, we summarize the most relevant challenges that we try to address in this dissertation.

1.2.1 Big data

Today's computation theory is based on the discretization of observable data into events associated to a finite time [Hog92]. Following this theory, even the eternity (*i.e.,* long term tasks) can be observed and computed as a sequence of finite temporal data points. Each observable event is then stored by augmenting data with a timestamp, for instance by creating a model element instance with an attribute timestamp [Pla]. The discretization process of a sensor can easily lead to millions of values in a short amount of time. The raise of IoT leads to a massive amount of data, which must be stored and processed. For this reason it is often associated with Cloud and Big Data technologies [GBMP13] ,[Pla]. However, the required storage and computation power to store and process this amount of data are a challenge, especially for stream and real-time analyses [HAG+13], which is the case for the IoT domain. Jacobs [Jac09] even calls the modeling of timed data as enumerations of discrete timed values, *Big Data pathology* or *Big Data trap*. Regardless of the chosen solution, intelligent systems need a structured representation of timed data in order to (periodically) reason about it.

1.2.2 Continuous aspect of physical measurements

The physical time is continuous and data splitting is only due to the clock-based conception of computers. IoT related data are collected value by value (due to regularly sampled measurements) but are virtually continuous. For example, for a temperature sensor, at any point in time it should be possible to extrapolate a value (even if there is no measurement at this specific point in time taken by the sensor). Indeed, the absence of a data record for a temperature value at a time t does not, of course, mean that there is no temperature at this time t . It could simply be that the sensor is saving energy by reducing its sampling rate or it is just not fast enough to measure the temperature at the requested granularity. The last measured temperature, recorded for time $t - 1$, would be the closest value to the requested one, because the physical value in reality is continuous. However, when there is a network loss of values, this approximation does not always work, if the signal is changing fast.

1.2.3 Inaccuracies and loss of values

Measurement inaccuracies are an inherent part of all physical measurements and strongly considered in signal processing techniques. Moreover the lack of processing power or memory or troubles in network communications can lead to loss of data values. These faults can pollute the data and mislead the reasoning afterwards. Neither the continuous nature of physical measurements nor measurement inaccuracies or network losses are currently considered in modeling techniques.

1.2.4 Distributed, dynamic and heterogeneous

As sensors and devices are getting more processing, storage and network capabilities, any reasoning process can be distributed on these devices, in order to offer more resilient AmI services. Centralized storage and processing lead to a bottleneck and is not suitable for large IoT systems. Moreover, if the network connectivity fails for any reason in the central node, the whole functionality of the system is threatened. Distributed reasoning allows IoT systems to be more resilient to data loss, network failures and takes advantage of all the available resources. The dynamic aspect comes from the fact that some devices might not be connected all the time, for instance when they are in power saving mode, or in the case of smartphones when they are unreachable due to a lack in connectivity. At any moment in time, some devices might get connected and other disconnected. This should be taken into account when building a reasoning platform on top of a continuously changing infrastructure. Another consideration to take into account is the heterogeneous nature of the different devices: each one can have different constraints in term of computation, memory, storage and power usage.

1.2.5 Context awareness

AmI systems are expected to support humans in their every day tasks and activities in a personalized fashion. This personalization depends on contextual information that can be derived from sensors raw data. Context-aware computing is one solution that allows to store context information linked to sensor data besides raw data so that the interpretation can be done easily and more meaningfully. Taking the previously enumerated challenges into consideration, context awareness is difficult to achieve in a distributed, dynamic and heterogeneous way.

Moreover, the imperfect nature of the derived context, and the special characteristics of AmI environments impose several challenges. Henricksen and Indulska [HI04] characterize four types of imperfect context: *unknown*, *ambiguous*, *imprecise*, and *erroneous*. Sensor or connectivity failures, result in situations where not all context data is available at any time. When data about a context property comes from multiple sources, then context may become ambiguous. Imprecision is common in sensor-derived information, while erroneous context arises as a result of human or hardware errors. Due to the highly dynamic and open nature of the environments and the unreliable wireless communications, agents do not typically know *a priori* all other entities present at a specific time, nor can they communicate directly with all of them. Despite these restrictions, they must be able to reach common conclusions about the state of the environment and collectively take context-aware decisions.

1.2.6 Adaptability

With dynamic environments and changing contexts in mind, it is important that applications support some degree of adaptability. Hence, up-to-date information about the user, available

services and host platforms, network connectivity, time, location and other sensed data can trigger appropriate application adaptation. Depending on the context, different system qualities need to be satisfied to a different extent. For example, in the AAL domain and in a normal context, privacy and security are highly important but they can be relaxed in an emergency situation. For instance a doctor or an ambulance are allowed to access the sensed raw values in this case. Another example: in the context of a device running low in battery, it is better to do heavy calculations on a remote server rather than locally and further drain the battery. Adapting these system qualities often involves optimizing conflicting objectives in order to find a good trade-off, and this optimization needs to be executed each time the context changes, or whenever the target objectives change for the same context.

1.2.7 Unobtrusiveness

One important goal of ambient intelligence as presented previously is to make computers not intrusive. This goal cannot be achieved, if for every context change, the user is asked to perform some actions or take some decisions manually over and over again. To make AmI systems easily adopted, they need to run with minimum configurations from the user side, in a seamless and unobtrusive way, while at the same time allowing users to interact when needed. Machine learning techniques such as clustering, classification, prediction, and probabilistic modeling offer great mathematical tools to learn user patterns and habits, to identify the current context, or to recommend and prioritize system qualities for each different context. However, most machine learning techniques work in batch processing mode and are not adapted to the dynamic and continuously changing aspect of AmI environments.

1.3 Contributions

Before presenting the contributions of this thesis, we introduce the related background needed to understand the main concepts in this dissertation. In chapter 2, we show the different types of contextual reasoning. After that, we present the ideas behind model driven engineering and models@run.time, in chapter 3, that will allow us to abstract and represent contextual information and create a distributed component models for AmI systems. Then we present the main qualities that need to be satisfied for AmI systems in chapter 4. We end up the background section with an overview on reasoning techniques, mainly the multi-objective optimization algorithms and machine learning techniques, in chapter 5.

In this thesis, we present the following contributions:

First, in chapter 6, we introduce an efficient way to encode physical sensor data in a continuous representation, adapted for AmI, resilient to data loss and which can extrapolate the missing values, in case of a network loss. We consider AmI agents as the smallest atomic distributed reasoning elements, having minimum connectivity, storage, computation and reasoning capabilities. Each AmI agent has a local knowledge base containing the

information gathered from one or several sensors connected to it as presented in figure 1.1. Our contribution enables the agents to keep a compact model of sensor data and retrieve approximated information from the past in a fast and efficient way even when data is missing.

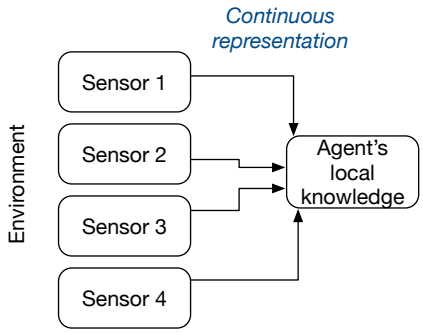

Figure 1.1: Local knowledge of an AmI agent.

In a next step, we consider that the agents have a remote and dynamic knowledge base about other connected agents and the information they can provide, plus a knowledge base containing context rules that allows to derive the current context. We assume first that the context rules are manually filled. We present a solution to enable context awareness in this distributed environment. Our solution allows any agent to query any other connected agent to get remote information needed to derive the context while handling inconsistencies, failures and potential query deadlocks. The process is presented in more details in chapter 7 and illustrated in figure 1.2.

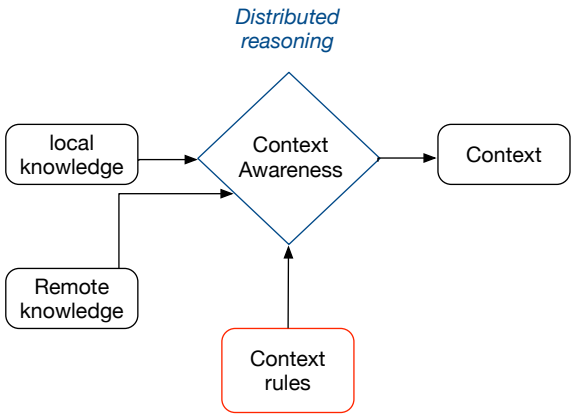

Figure 1.2: Distributed context awareness.

Whenever the context changes, it will trigger an internal adaptation process in order to find the best trade-off between several qualities defined for the new context. The adaptation

process relies on a multi-objective evolutionary algorithms (MOEA) that we adapt to run directly on top of models as it will be presented in chapter 8. In order to achieve the privacy quality, we propose to use **blurring components** as our main privacy preservation elements. The idea behind this approach, as presented in chapter 9, is that, by gradually decreasing the data quality, a blurring component is able to hide some of the personal data delivered by sensors while still keeping necessary information for the services to work. However, these components act as well on other orthogonal qualities such as utility and efficiency. Combining MOEA and blurring components, we enable an adaptable platform that can find good trade-offs between the different qualities according to the current context of AmI. This is presented in 1.3.

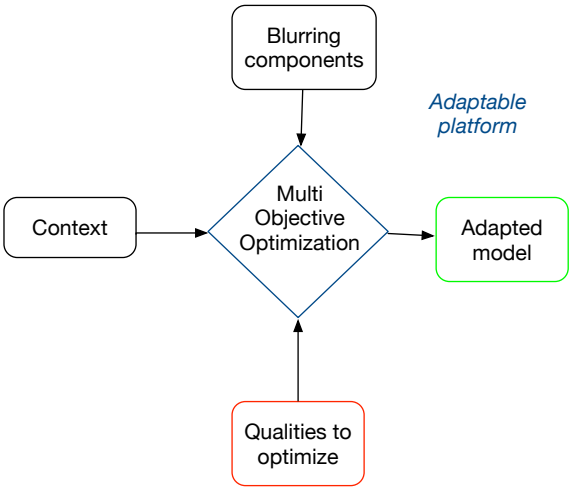

Figure 1.3: Adaptable platform

Finally, in order to achieve the unobtrusiveness objective, the context derivation rules and the qualities to optimize for every context need to be learned automatically with the least intervention from the user. This dissertation ends up by opening currently ongoing research directions on how to model and adapt machine learning techniques to fit for AmI platforms. These techniques offer powerful tools to automatically detect patterns, categorize contexts, build usage profiles and provide statistical information vital for the intelligent aspect of AmI. When integrated to AmI solutions, these techniques will respond to the unobtrusiveness challenge. They will be able to make the platform learn from users habit silently with minimum interruption.

To summarize, here is the following contributions in this thesis, put together in a big picture (figure 1.4). The boxes in red are assumed to be filled manually in a first step. In the last contribution chapter we discuss how these boxes can be inferred automatically.

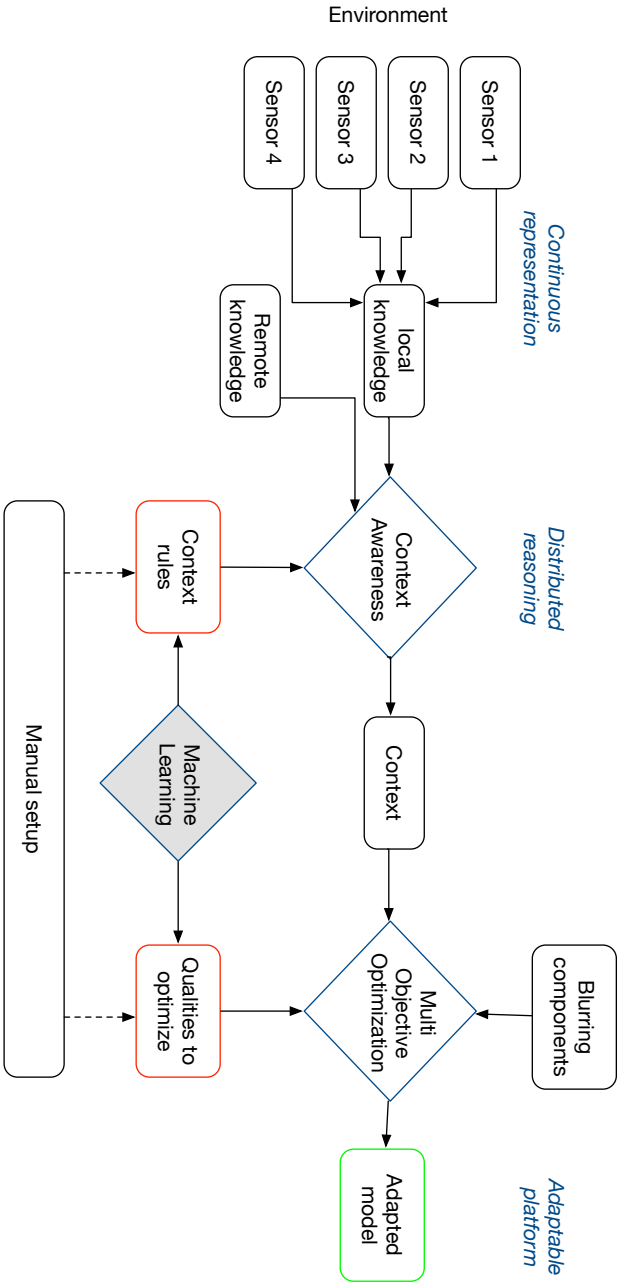

Figure 1.4: Contributions of this thesis, the big picture.

Part I

Background and state of the art

2

Contextual reasoning

A context is any information that can be used to characterize the situation of the environment, a person, a place, or any object that is considered relevant to the interaction between the user and the application, including the user and application themselves. Context-aware AmI systems use context information in order to adapt their operations without an explicit user intervention and thus aim at increasing usability and effectiveness providing better and more adapted services to the user. In this chapter, we present the formal definitions of contexts and contextual reasoning. We present as well the state of the art of the different contextual reasoning platforms and architectures.

This chapter is based on the work that has been published in the following papers:

- Assaad Moawad, Antonis Bikakis, Patrice Caire, Grégory Nain, and Yves Le Traon. R-core: A rule-based contextual reasoning platform for ami. In *RuleML@ ChallengeEnriched 2013*, 2013
- Assaad Moawad, Antonis Bikakis, Patrice Caire, Grégory Nain, and Yves Le Traon. A rule-based contextual reasoning platform for ambient intelligence environments. In *Theory, Practice, and Applications of Rules on the Web*, pages 158–172. Springer, 2013

Contents

2.1 Definition

With the appearance of mobile devices such as notebooks, PDAs, and smartphones, pervasive (or ubiquitous) systems are becoming increasingly popular these days. Appliances should vanish into the background to make the user and his tasks the central focus rather than computing devices and technical issues. One field in the wide range of pervasive computing are the so-called context-aware systems. Context-aware systems are able to adapt their operations to the current context without explicit user intervention and thus aim at increasing usability and effectiveness by taking environmental contexts into account. Particularly, when it comes to using mobile devices, it is desirable that programs and services react specifically to their current location, time and other environment attributes and adapt their behavior according to the changing circumstances as context data may change rapidly [BDR07]. The needed information may be retrieved in a variety of ways, such as applying sensors, network information, device status, browsing user profiles and using other sources.

The history of context-aware systems started when Want et al. (1992) [WHFG92] introduced their active badge location system, which is considered to be one of the first context-aware applications. Over the last decade, location information became one of the most frequently used attribute of context.

In the literature, one of the first occurrences of the term context-aware appeared in Schilit and Theimer (1994) [ST94]. There the authors describe context as location, identities of nearby people, objects and changes to those objects. Such enumerations of context examples were often used in the beginning of context-aware systems research. In 1999, Dey et al. [ADB+99] suggest the following types, dimensions or categorizations of contexts: **location, activity, identity and time**. Moreover, they define context as follows:

Definition 1 *"Context is any information that can be used to characterize the situation of entities (i.e., whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves."*

In this thesis, we put the notion of context at the center of AmI solutions. On one side, data collection from sensors are used in order to derive this contextual information. On the other side, once the context is derived, it is used to reason and adapt the AmI platform accordingly. However, representing and deriving the context in AmI is not an easy task. This is mainly due to the continuous aspect of the physical sensed attributes, and due to the distributed and heterogeneous nature of AmI agents.

2.2 Architecture

Several approaches exist to achieve context-aware systems. These approaches depend on the requirements and conditions of the system, such as the location of sensors (local or

remote), the amount of possible users (one user or many), the available resources of the used devices (high-end-PCs or small mobile devices). Chen et al. [CFJ03] presents three different approaches on how to acquire contextual information:

1. **Direct sensor access**: This approach is often used in devices with sensors locally built-in. The client software gathers the desired information directly from these sensors, i.e., there is no additional layer for gaining and processing sensor data. Therefore, it is not suited for distributed systems due to its direct access nature, which lacks a component capable of managing multiple concurrent sensor accesses.

2. **Middleware infrastructure**: Modern software design uses methods of encapsulation to separate e.g., business logic and graphical user interfaces. The middleware based approach introduces a layered architecture to context-aware systems with the intention of hiding low-level sensing details.

3. **Context server**: The next logical step is to permit multiple clients access to remote data sources. This distributed approach extends the middleware based architecture by introducing a component managing remote access. Gathering sensor data is moved to this so-called context server to facilitate concurrent multiple access. Besides the reuse of sensors, the usage of a context server has the advantage of relieving clients of resource intensive operations.

2.3 Contextual reasoning challenges in AmI

The aim of reasoning about context in ambient intelligent systems is to exploit the true meaning of raw context data; to process, combine and ultimately translate the low-level data of the sensors into valuable information, based on which the system can determine the state of its context, and react appropriately to certain context changes.

The uncertainty and imperfection of context information, and the special characteristics of the entities that operate in ambient intelligent environments introduce, however, several challenges in this task. Sensor or connectivity failures (which are inevitable in wireless connections) result in situations, that not all context data is available at any time. When data about a context property comes from multiple sources, then context may become ambiguous. Imprecision is common in sensor-derived information, while erroneous context arises as a result of human or hardware errors. The entities that operate in an ambient intelligent environment are expected to have different goals, experiences and perceptive capabilities. They may use distinct vocabularies. Due to the highly dynamic and open nature of the environment, various entities join and leave the environment at random times. Moreover, they unreliable and restricted by the range of the transmitters or wireless communications. Ambient agents do not typically know *a priori* all other entities that are present at a specific time nor can they communicate directly with all of them [Bik09].

Overall, according to [Bik09], the role of reasoning about context in ambient intelligent systems includes:

1. Detecting possible errors in the available context information;
2. Handling missing values;
3. Evaluating the quality and the validity of the sensed data;
4. Transforming the low level raw context data into higher level meaningful information so that it can later be used in the application layer;
5. Making decisions about the system behavior when certain changes are detected in the system's context.

Considering these special characteristics of context reasoning in AmI, the three main challenges, according to [Bik09], are to enable:

1. Reasoning with the highly dynamic and imperfect context.
2. Managing the potentially huge piece of context data, in a real-time fashion, considering the restricted computational and storage capabilities of some mobile devices, and the constraints imposed by wireless communications.
3. Collective intelligence, by supporting heterogeneous information sharing, and distributed reasoning with all the available context information.

2.4 Multi-Context System MCS

In ambient intelligent systems, each agent can have its own set of possible contexts depending on its application domain. Multi-context systems (MCSs) define a versatile framework for integrating and reasoning about knowledge from different (heterogeneous) sources.

A simple and classic illustration of the intuitions underlying multi-context systems is provided by the so-called 'magic box' example, illustrated in figure 2.1:

When Mr.1 and Mr.2 look at a box, which is called "magic" because neither of the observers can make out its depth, both Mr.1 and Mr.2 maintain a representation of what they believe to be true about the box. Mr.1's beliefs may regard concepts that are meaningless for Mr.2, and vice versa. Another important notion is that the observers may have (partial) access to each other's beliefs about the box.

Reasoning with multiple contexts is a combination of:

- Local reasoning, which is performed using knowledge of a single context. Local conclusions in a given context derive from a set of axioms and inference rules that model local context knowledge, and which constitutes a small subset of the entire knowledge.

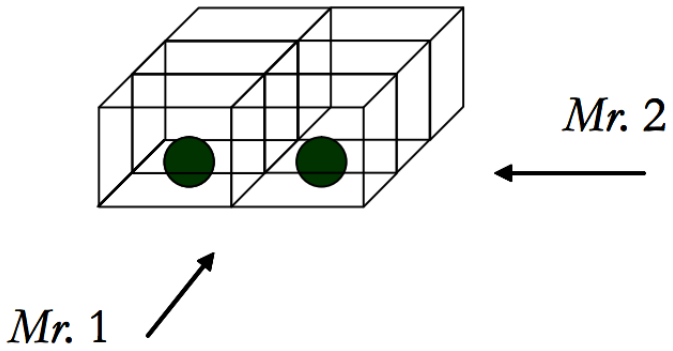

Figure 2.1: A magic box example

- Distributed reasoning, which also takes the possible relations between local contexts into account. These relations result from the fact that different contexts actually constitute different representations of the same world. They are modeled as inference rules (known as mapping or bridging rules) with premises and consequences, and enable an information flow between related contexts.

In the domain of multi-context modeling, several techniques are possible. The survey done by Parera et al. [PZCG14] enumerate at the following possible techniques: Key-value, markup scheme tagged encoding, object based, logic based, or ontology based. They enumerate as well the pros and cons for each of these techniques. Comparing with the requirements of ambient intelligent systems, especially the need to be reactive in near-real time, and the ability to be distributed, the logic base modeling approach is suitable for our application domain, specifically the modeling based on defeasible logic.

2.5 Contextual Defeasible Logic

Contextual Defeasible Logic (*CDL*) is a distributed rule-based approach for contextual reasoning, which was recently proposed as a reasoning model for ambient intelligent systems [BA10]. CDL adopts ideas from:

- *Defeasible Logic* [ABGM01]: it is rule-based, skeptical, and uses priorities to resolve conflicts among rules.

- *Multi-Context Systems* (MCS, [GS94, GG01]): logical formalizations of distributed context theories connected through mapping rules, which enable an information flow between contexts. In MCS, a *context* can be thought of as a logical theory - a set of axioms and inference rules - that models local knowledge.

In the next section, we present the representation model of CDL and explain how it fits with the special characteristics and requirements of ambient intelligence environments.

2.6 Contextual Representation Model

In CDL, the MCS model is extended with defeasible rules and a preference relation on the system contexts. Let's consider C as the set of contexts C_i in a MCS: A context C_i is defined as a tuple of the form (V_i, R_i, T_i), where V_i is the vocabulary of C_i, R_i is a set of rules, and T_i is a preference ordering on C.

V_i is a set of positive and negative literals of the form $(c_i : a_i)$ and $\sim (c_i : a_i)$, which denotes the negation of $(c_i : a_i)$. Each context uses a distinct vocabulary, i.e., $V_i \cap V_j = \emptyset$ *iff* $i \neq j$. This reflects the fact that each entity (e.g. device in an ambient intelligent environment) may use its own terminology. We should note, though, that the proposed model may also enable different contexts to use common literals by adding a context identifier, e.g. as a prefix, in each such literal and using appropriate mappings to associate them to each context.

R_i consists of a set of *local rules* and a set of *mapping rules*. The body of a local rule is a conjunction of *local* literals (literals that are contained in V_i), while its head is labeled by a local literal. There are two types of local rules:

- Strict rules, of the form

  $$r_i^l : (c_i : a^1), \ldots, (c_i : a^{n-1}) \rightarrow (c_i : a^n)$$

  They express sound local knowledge and are interpreted as follows: whenever the literals in the body of the rule $((c_i : a^1), \ldots, (c_i : a^{n-1}))$ are strict consequences of the local theory, then so is the conclusion of the rule $((c_i : a^n))$. Strict rules with empty body denote factual knowledge.

- Defeasible rules, of the form

  $$r_i^d : (c_i : a^1), \ldots, (c_i : a^{n-1}) \Rightarrow (c_i : a^n)$$

  They are used to express uncertainty: a defeasible rule cannot be applied to support its conclusion if there is adequate contrary evidence.

Mapping rules associate local literals of C_i with literals from the vocabularies of other contexts (*foreign literals*). The body of each such rule is a conjunction of local and foreign literals, while its head is labeled by a local literal. Mapping rules are modeled as defeasible rules of the form:

$$r_i^m : (c_j : a^1), \ldots, (c_k : a^{n-1}) \Rightarrow (c_i : a^n)$$

A mapping rule associates literals from different contexts (e.g. $(c_j : a^1)$ from C_j and $(c_k : a^{n-1})$ from C_k), with a local literal of the context that has defined r_i, which labels the head of the rule (here $(c_i : a^n)$). By representing mappings as defeasible rules, we can deal with ambiguities and inconsistencies caused by importing mutually conflicting information from different contexts. Finally, each context C_i defines a strict total preference ordering T_i

on C to express its confidence in the knowledge it imports from other contexts:

$$T_i = [C_k, C_l, ..., C_n]$$

C_k is preferred to C_l by C_i, if C_k precedes C_l in T_i. The strict total ordering enables resolving all potential conflicts that may arise from the interaction of contexts through their mapping rules. In a later version of CDL [BA11], T_i is defined as a partial preference order on C, which enables handling incomplete preference information. For sake of simplicity, we adopt here the original definition of T_i. In this thesis, we adopt ideas from contextual defeasible logic in order to develop a context aware framework suitable for ambient intelligent systems. This will be presented in more details in chapter 7.

3

Modeling frameworks

The distributed and heterogeneous nature of ambient intelligent systems lead to an important complexity when it comes to representing their current context or state, in order to later reason about it. To cope with such complexity, an abstract layer is needed to represent the main domain concepts, and the relationship between them. For this goal, various methods such as model-driven engineering can help to structure and organize the amount of data in order to process it for a later reasoning. In this chapter, we introduce the main concepts of model-driven engineering, frameworks and features. More specifically, we focus on the models@run.time paradigm which is tailored for context-aware and adaptive systems. We first introduce the Eclipse Modeling Framework with its advantages and limitations. We then present the Kevoree Modeling Framework and show its suitability for the AmI domain. Finally, we present Kevoree [FDP+12], the component-based platform that we use throughout this thesis as our main development tool, we we integrate prototypical implementation of the concepts developed in this thesis.

Contents

3.1 Introduction

AmI systems aim to "make the system adapt to people," as opposed to "people adapting to the system". In order to achieve this, context awareness is a first step needed to trigger system adaptations. However, the distributed and heterogeneous nature of ambient intelligent systems lead to an important complexity when it comes to representing their current context or state, in order to later reason about it. To cope with such complexity, an abstract layer, that is always synchronized with the reality, is needed to represent the main domain concepts, and the relationship between them. For this goal, various methods such as model-driven engineering can help to structure and organize the data, in order to process it for a later reasoning. This step requires abstracting the system into domain specific concepts and relationships between these concepts to ease context derivation and the reasoning afterwards. For this goal, we use modeling techniques from model-driven engineering that we will present in section 3.2. For every AmI agent, we use data models to represent the knowledge this agent possess. For the whole AmI system, we use distributed component models, where every component is an abstraction for an AmI agent. The next step is to enable a reflection layer where we feedback the system representation to the system itself to enable self-adaptation. In order to achieve this, we use models@run.time which we present in section 3.3.

3.2 Model-driven engineering

Model-driven engineering (MDE), aims to raise the level of abstraction in application domains in order to extract and capture more easily the semantics and concepts used in this domain and the relationships between these concepts. MDE technologies provide a variety of promising approaches to address platform complexity and express domain concepts effectively, by combining the following [Sch06]:

- **Domain-specific modeling languages**: DSML aim at providing domain-specific developers with dedicated languages and tools specifically created and tailored for their business domain, such as software-defined radios, avionics mission computing, online financial services, warehouse management, or even the domain of middleware platforms [Sch06]. DSMLs usually leverage an object-oriented software API, generated from a metamodel which define the relationships among concepts in a domain and precisely specify the key semantics and constraints associated with these domain concepts. This technique facilitates the authoring and exchange of domain-specific concepts between several tools, applications and stakeholders. This API can for example support graphical or textual editors to edit models, or specific load and save algorithms to serialize and exchange models. The primary focus of DSMLs is to ease the comprehension and manipulation of concepts during the design phase of an application. In this area, the Eclipse Modeling Framework (EMF) has rapidly become

the *de facto* standard, in MDE for building domain specific models and associated tools using generative techniques.

- Transformation engines and generators: that analyze certain aspects of models and then synthesize various types of artifacts, such as source code, simulation inputs, or alternative model representations [Sch06]. This ability to create artifacts from models helps ensure the consistency between application implementations and analysis information associated with the requirements captured by models. Moreover, MDE tools can impose domain-specific constraints and perform model checking that can prevent many errors from happening early in the life cycle of software development.

In this thesis, we use two kinds of models: architectural models and data models. Architectural models to represent what components are present in the current AmI system, and the connectivity between them (sensors, agents, communication channels). While data models manage the attributes offered by each component, within a time dimension. The data models should reflect as well the continuous aspect of the physically sensed attributes. These models need to be adaptable at runtime when the context changes, for this reason, we present the models@run.time paradigm in the next subsection.

3.3 Models@run.time for context representation

The models@run.time paradigm promises a new approach to MDE, by fading the boundary between design-time (the typical phase where MDE is employed) and runtime. These models@run.time[MBNJ09] are abstract, yet always synchronized, representations of the running system. Any change of the abstraction implies an adaptation of the system at runtime. Conversely, a dynamic adaptation triggered by the runtime is reflected as well in the model. This abstract model, always synchronized with the running system, enables local or remote computations on the system. It can be used to add or remove a component, or to redeploy a distributed application to do load balancing.

The dynamic models@run.time platform follows the idea to keep at runtime (during the execution of a system) a model of the current running system as a reflection layer [DFB+12]. This reflexion model allows not only to introspect (analyze) the current state, but also acts as an intercession layer (ability to modify a system through model modifications). In order to ensure this property, a bidirectional causal link [MBNJ09] has to be built, which impacts the system at each modification applied on the model and vice-versa. Any change in the system appears in the model, and vice versa, any change in the model affects the running system. The synchronization between the real world and its abstraction needs to be fast enough to call the AmI system responsive in near real-time.

A typical usage of models@run.time is to manage the complexity of dynamic adaptation in complex, distributed and heterogeneous systems, by offering a more abstract and safer abstraction layer than reflection on top of the running system [FNM+12b]. In this thesis, we consider models@run.time as the cornerstone to build applications for AmI, in order to keep the state of AmI system in sync with its abstraction model, in near real-time.

3.3.1 Requirements of models@run.time

The heterogeneity and distribution aspect creates specific requirements for models@run.time to be suitable to use in ambient intelligent systems. We present here these main requirements as listed by Fouquet et al. [FNM+12b]:

- **Reduced memory footprints**: The memory footprint of a models@run.time engine determines the types of nodes that are able to run this engine. The more demanding is the models@run.time engine in terms of memory, the more difficult it is to deploy it on the small devices (e.g. Android phones, gateways with low power CPUs), and the more centralized the application will be. This would reduce the reliability of the system: if the large devices fail or disconnect, the overall system cannot safely adapt anymore. Moreover, model exchanges for the synchronization of the system in this strategy would dramatically increase network load.
- **Thread safety**: A model@run.time is generally used in highly concurrent environments. For instance, different probes integrated in a device can update a context model. This model is then used for triggering the adaptation reasoning process. This context model should enable safe and consistent read and write for the reasoners to take accurate decisions. The model@run.time infrastructure must ensure that the multiple threads of the application can access and modify the models without worrying about the concurrent access details. It must implement fast, yet safe, validation or reasoning algorithms on multi-core/thread nodes.
- **Efficient model cloning**: A device should be able to locally clone its own model for reasoning purposes so that it can reason on a fully independent and safe representation of itself, which can later on be re-synchronized with the current model. This criteria is very important for reasoning on models, where we want to try several variations on the current model (for instance, in an optimization problem execution).
- **Lazy loading**: it is a design pattern that defer the initialization of a model until the point at which it is needed. Lazy loading can improve the performance and efficiency of AmI systems in many levels, for example: initialization time, memory consumption and network traffic are kept as low as possible, due to the fact that only the useful parts of the model are loaded and only when needed.

3.3.2 Native Independent Versioning

Data composing the reasoning context of intelligent systems can have very different update rates. For instance, a topology related information can be updated every month whereas a sensor related data will be updated every millisecond. The development of such systems, driven by models@run.time, implies an efficient strategy to manage these very volatile data together in a context model with static data. Indeed, saving the whole model for each modification offers a very poor performance, whereas creating timestamped elements for each change also leads to large and unmanageable models.

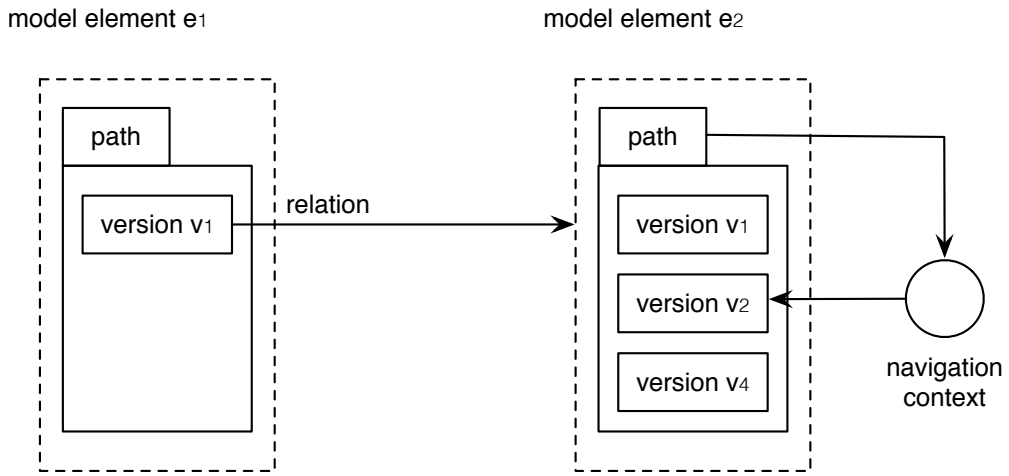

Figure 3.1: Native versioning for model elements

To tackle these problems, Hartmann et al. [HFN+14a] proposed, a native versioning concept for each model element. They defined that every model element can exist in a sequence of versions, attached to its unique identifier. They also defined model element relationships as links between unique identifiers, without the version information. They presented as well a navigation concept that always resolves the *closest* version to the one requested. For instance, if the requested version is $v3$, the navigation context will resolve the closest one $v2$ in case $v3$ does not exist. This is illustrated in figure 3.1.

3.3.3 Time management

This model version can be used to simulate time continuity. For instance, at each attribute change, a new version of the attribute is created; and the current time can be used as the version ID. This method enables to model classical discrete time-series using the native independent versioning solution for each granular update. The time-continuity is simulated by the navigation context resolver which takes the closest version of the requested one [HFN+14b]. For example, if the sensor has pushed values at time t and $t + 100$, any get request on the attribute between t and $t + 100$, let's say at $t + 50$, will resolve the version stored at time t. However, this creates a discontinuity on the attribute-value dimension. A linear signal for example will be transformed to steps-shaped signal. However, fo physical attributes, we would expect that the value to change smoothly from time t to $t + 100$. Moreover, based on this discrete representation of versions, this approach still expects the enumeration capabilities of versions and thus is limited to face a high sampling rate from a sensor. For instance, at a high sampling rate of a constant signal, each sample will be saved in a new version, even if the value is still the same. The tree that manages the version resolution will increase in size a lot, affecting the performance of a lookup search. For these reasons, we propose an efficient and continuous model for IoT in chapter 6.

3.3.4 Eclipse Modeling Framework (EMF)

EMF [SBMP08] is a modeling framework and code generation facility for building tools and other applications based on a structured data model. From a metamodel specification described in ECORE, EMF provides tools and runtime support to create Domain Specific Languages (DSL) on top of the Eclipse platform. It produces a set of Java classes for the model, along with a set of adapter classes that enable viewing and command-based editing of the model, and a basic editor. Most important, EMF provides the foundation for interoperability with other EMF-based tools and applications.

However, EMF suffers from several limitations that make it not suitable for models@run.time. First, it has a large dependency set. In [FNM+12b], it has been shown that a standalone JAR executable outside of the Eclipse tool (a Java archive that includes all dependencies) has a size of 15 MB for only 55 KB generated files. The large number and size of dependencies is one of the main limitations of EMF when the model must be embedded at runtime. Second, EMF does not provide thread safe accesses to the models [GvdHT09]. This requirement is important for a model@run.time infrastructure, because supporting dynamic and distributed architectures requires concurrent access to models. A third limitation, is the cloning overhead. As reported in [FNM+12a], EMF has a large memory footprint when cloning a model, and it is slow. Forth, EMF has a poor performance when loading large models, and finally EMF does not manage time natively. For these limitations, we decided to use Kevoree Modeling Framework (KMF) instead of EMF because it is more suitable for the needs of AmI.

3.3.5 Kevoree Modeling Framework (KMF)

The Kevoree Modeling Framework or KMF [i], is an open source alternative to EMF. KMF was made to support a generic and efficient models@run.time infrastructure compatible with EMF. KMF addresses the limitations of EMF by offering the following features.

- **Multi-thread access**: In order to allow multi-threading, KMF deploys a protection against concurrent read and write accesses.
- **Fast cloning and lazy loading**: KMF has internal mechanisms to clone models fast, and in an efficient way (Refer to table 3.1 for a comparison between KMF and EMF [FNM+12a]). The fast cloning will allow us to execute mutations on models faster, when performing optimizations. This will be discussed in more details in chapter 8.
- **Time-aware model**: KMF treats time dimension as a crosscutting concern of data modeling. Indeed, model elements can independently evolve in time at different paces, so there is no need to enforce the sampling of all model elements at the same rate. KMF does not store snapshots of the entire model but rather only stores updated

[i] http://kevoree.org/kmf/

model elements together with their relative time [HFN+14b]. Moreover, it provides a natural and seamless navigation concept to navigate into the space-time continuum of data. Most importantly, KMF enables a model to combine elements from different points of time, forming a time-distorted model, which is especially useful for (near) real time reasoning requirements [HFN+14b].

In order to be able to handle big amounts of continuous data coming from physical sensors, and in order to be able to represent the time continuity, we extend KMF by introducing a new meta attribute type for models@run.time, which can dynamically encode continuous IoT data, as a first contribution of this thesis. KMF will be used for data models throughout the thesis.

Table 3.1: Performance comparison between EMF and KMF

Feature	EMF	KMF	Comparison
Model creation	376 ms	313 ms	1.2 times faster
Model clone	3588 ms	398 ms	9 times faster
Model save	7021 ms	2630 ms	2.66 times faster
Heap memory footprint	104MB	61MB	1.70 times lighter

3.4 Kevoree - A component based software platform

In Ambient Assisted Living (AAL), systems need to combine various data from various agents to reason about it. These agents are distributed and heterogeneous (meaning that each one can be running on a different technology). Thus there is a need to use a distributed component based software platform that acts as a middleware and offers a communication layer between these agents. *Kevoree* [FBP+12] is an open-source environment that provides means to facilitate the design and deployment of distributed dynamically adaptive systems, taking advantage of model@run.time mechanisms throughout the development process [MBNJ09] .

This development platform is made of several tools, among others the Kevoree Modeling Framework (KMF) (that we previously introduced), a model editor (to assemble components to create an application), and several runtime environments, from Cloud to JavaSE or Android platforms. Kevoree is build on top of a component model and aims to enable distributed and reconfigurable software development. The component model of Kevoree defines several concepts. These concepts are dynamically modifiable through models@run.time. In other words any model-based process could drive the platform in terms of adaptations.

The **Node** (in grey in figure 3.2) is a topological representation of a Kevoree runtime. There exist different types of nodes (e.g.: JavaSE, Android, etc.).

Component instances are deployed and run on a node instance, as presented on figure 3.2. Components may also be of different types, and one or more, heterogeneous or not,

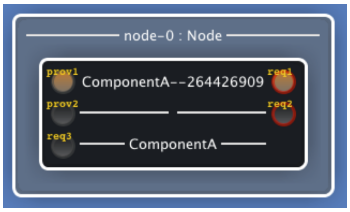

Figure 3.2: A component instance, inside the node instance on which it executes

component instances may run on a single node. Components declare **Ports** (rounds on left and right sides of the component instance) for provided and required services, and input and output messages. The ports are used to communicate with other components of the system.

Groups are used to share models (at runtime) between execution platforms (i.e. nodes). There are different types of Groups, each of which implements a different synchronization / conciliation / distribution algorithm. As the shared model is the same for all the nodes, there may be some concurrent changes on the same model, that have to be dealt with.

Finally (for the scope of this thesis), **Channels** handle the semantics of a communication link between two or more components. In other words, each type of channel implements a different means to transport a message or a method call from component A to component B, including local queued message list, TCP/IP socket connections, IMAP/SMTP mail communications, and various other types of communication.

3.5 Kevoree Critical Features for AmI

Ambient intelligence systems are composed of several entities called agents. These agents can be autonomous or work collaboratively to achieve a given task. They are provided with different capabilities, making one more able to make huge computations while others work better as communication relays. Dynamicity plays an important role in ambient intelligent systems, because agents can appear and leave in an unpredictable way. For instance, a smart phone will join a system when a person enters a room, and leave as soon as this person is too far from the room. Kevoree has been designed to ease the development of ambient intelligence systems. It is an appropriate choice to provide solutions to the dynamic nature of such systems, in which agents are often autonomous, reactive and proactive in order to collaborate and fulfill tasks on behalf of their users.

In Kevoree, an agent is represented as a node that hosts one or more component instances. A synchronization group is responsible for the communication with other nodes. Some group types implement algorithms with auto-discovery capabilities, making nodes and their components dynamically appear in the architecture model of the overall system. The fact that a new node appears in the model means that an agent is reachable, but it does not necessarily mean that it participates in any interaction. The component instances of a node provide the services for the agent. Therefore, for an agent to take part in a collaborative

work, the ports of the component instances it hosts have to be connected to some ports of other agents' components.

Some features of Kevoree make it particularly suitable for our needs. First, it enables the implementation, deployment and management of heterogeneous entities as independent *nodes*. Second, it uses communication *channels* to enable the exchange of messages among the distributed components. Third, it offers a common and shared representation model for different types of nodes. Finally, it is endowed with adaptive capabilities and auto-discovery, which fit with the open and dynamic nature of AmI environments. Kevoree will be used for components models throughout the thesis.

4

Qualities in AmI

When the context changes, an AmI agent needs to readapt accordingly. The adaptation is needed in order to satisfy the qualities required for the new context. In this chapter, we list the several qualities that are related to the AmI domain, mainly: privacy, utility and efficiency. For every quality, we will provide a definition and an overview of the state of the art.

A big part of this chapter will be published in the following journal paper, which is a systematic review about privacy definitions and challenges in ambient intelligent systems.

- Efthymiou Vasilis Bikakis Antonis Caire Patrice, Moawad Assaad and Le Traon Yves. Privacy challenges in ambient intelligence systems. *Journal of Ambient Intelligence and Smart Environments*, 2016

Contents

4.1 Introduction

When the context changes, an AmI agent needs to re-adapt accordingly. The adaptation is needed in order to satisfy the qualities required for the new context. In previous research papers, the term quality is also known as non-functional requirement. For a definition of quality, an IEEE standard [23793] is given here:

Definition 2 *"Software quality is the degree to which software possesses a desired combination of attributes (e.g., reliability, interoperability)."*

Colloquially speaking, qualities have been referred to as well as "-ilities" (e.g., usability) or "-ities" (e.g., integrity), i.e., words ending with the string "-ility" or "-ity. A large list of such words can be found, for example, in [CNYM12]. There are many other types of qualities that do not end with these terms as well, such as privacy, userfriendliness, performance and coherence.

In AmI systems, qualities are contextual. Meaning that, for every context, qualities can be satisfied to different extents according to their priorities for the specific context. For instance, in a normal situation, a health care agent is expected to operate with a high level of privacy. However, in an emergency situation, the level of privacy can be decreased for a trusted neighbor, a doctor, or the ambulance, in order to provide medical assistance (utility). In this chapter, we present three main qualities that we consider important in the AmI and AAL domain: privacy, utility and efficiency. For every quality, we provide a definition and an overview of the state of the art. We focus heavily on the privacy aspect, due to the fact that one of our main application domains in the CoPAInS project (Conviviality and Privacy in Ambient Intelligent Systems) is AAL. In this domain, privacy aspects play an important role for acceptance and usage of medical assistive technologies [WZ11].

4.2 Privacy

The notion of privacy has been discussed extensively, not only over the last decades, but even as early as the 19th century, and appears in the literature of various disciplines. However, there is no universal definition for privacy, and many researchers have referred to the difficulties involved in trying to produce such a definition [Bur82, BN98, LKVD+01]. Newell, for example, argues that this is due to the multidisciplinary nature of privacy, and refers to the difficulty of relating studies from different disciplines [BN98]. In this section, we will present first the several definitions of privacy, the different issues and the state of the art of privacy preservation techniques.

4.2.1 Definitions of privacy

A part of the systematic review we did about the state of the art of the privacy in AmI is to classify the different references we found defining privacy into three main categories: privacy as a right, as an enabler, and as a commodity. In this section, we present these definitions and their relations to the ambient intelligence domain.

4.2.1.1 Privacy as a Right

Warren and Brandeis [WB90] are usually credited the definition of privacy as "the right to be let alone", which was actually a reference to Thomas Cooley's "Treatise on the Law of Torts" [Coo78], written twelve years earlier, in 1878. The need for such a right emerged from the "unauthorized circulation of portraits of private persons", performed by the newspaper enterprises which used instantaneous photographs. This basic definition remained the most famous as shown in the article titled "One hundred years of privacy" [Gor92]. Some other relatively simple descriptions, include "exclusive access of a person to a realm of his or her own", or "control over information about oneself".

Solove [Sol06] presented a taxonomy of privacy to serve as a framework for the development of the field of privacy law. In this taxonomy, he classified harmful privacy-related activities into four groups: *i*) Information collection: surveillance, interrogation. *ii*) Information processing: aggregation, identification, insecurity, secondary use, exclusion. *iii*) Information dissemination: breach of confidentiality, disclosure, exposure, increased accessibility, blackmail, appropriation, distortion. *iv*) Invasion: intrusion, decisional interference. He presented each of these activities and focused on how they affect an individual's life in a harmful way, stressing on the importance of privacy as a right.

4.2.1.2 Privacy as an Enabler

A second category of definitions of privacy includes Alan Westing's definition as "the ability of people to determine for themselves when, how, and to what extent information about them is communicated to others" [Wes67] and Stefanos Gritzalis's as "the indefeasible right of an individual to control the ways in which personal information is obtained, processed, distributed, shared, and used by any other entity"[Gri04]. Moreover, Kupfer [Kup87] states that "privacy enables control over personal information as well as control over our bodies and personal choices for our concept of self", making privacy subjective to every person's own "concept of self".

Hong et al. [Hon04] point out that privacy is a fluid and malleable notion with a range of needs and trust levels rather than being a single monolithic concept. They focused on empowering people with choice and informed consent, letting individuals share personal information with the right people and services, in the right situations, and at the right level of detail.

Altman [Alt77] conceptualizes privacy as the "selective control of access to the self". Palen [PD03] argues that privacy management is not just about setting rules and enforcing them, rather than the continuous management between different spheres of action and degrees of disclosure within those spheres. DeCew [DeC97] suggests that privacy is a cluster concept covering interests in *i*) control over information about oneself, *ii*) control over access to oneself, both physical and mental, and *iii*) control over one's ability to make important decisions about family and lifestyle in order to be self-expressive and to develop varied relationships.

4.2.1.3 Privacy as a Commodity

Privacy seems to be a culturally relative right, but this doesn't mean that it is completely subjective [Moo03]. For instance, privacy is more and more considered to be a commodity in the US (since it relies on self-regulation) whereas in the EU, it is a human right. To bridge this gap and allow US companies to do business in the EU and conform to the EU Privacy Directive, EU and US arrived at an agreement, known as the Safe Harbor Agreement. This agreement offers a convenient way of complying with the adequacy requirements of the EU Directive [Ste02].

Privacy has nowadays become a commodity [Dav97], in the sense that the consumer makes a non-monetary exchange of their personal information for value such as higher quality service and personalized offers or discounts [CB03]. Consumers are becoming aware of the value of their personal data and are less and less ready to let businesses and companies, have this data without their explicit consent or even for free [DG03].

Tradeoffs must therefore be established between, on the one hand, the consumers who want to obtain goods and services [Pap10], and, on the other hand, businesses' eagerness to gather ever more knowledge and personal data on consumers to better target and streamline their consumer base. Such tradeoffs are therefore becoming part of the requirements of AmI systems.

A very recent work by Li et al. [LLMS13] presents a theory of pricing private data. They provide a framework to estimate monetary value of private information queried with a certain degree of accuracy. They define as well the term "privacy budget" which refers to a limit on the quantity and/or accuracy of queries that any buyer can ask, in order to prevent an unacceptable disclosure of the data for the owner.

4.2.1.4 Summary

Table 4.1 presents a categorization of the different approaches used to define privacy that have been discussed in this section. The table should be read as follows: We see privacy has been defined in three ways; as a right, as an enabler (to control personal data), and more recently as a commodity expressed through privacy policies of commercial products.

Right	Enabler	Commodity
-to be let alone [WB90]. -to be protected by law against the injury to the feelings, dignitary harm and reputational injury [Sol06]. -to keep a domain around us which includes all those things that are part of us, such as our body, home, property, thoughts, feelings, secrets and identity [OG+05].	-to determine when, how, and to what extent personal information is communicated to others [Wes67]. -to control the ways in which personal information is obtained, processed, distributed, shared, and used [Gri04]. -to empower people with choice and informed consent to share personal information with the right people and services, in the right situations, and at the right level of detail [Hon04]	-a non-monetary exchange of consumers' personal information for value such as higher quality service and personalized offers or discounts [CB03]. -to empower individuals to control their private data through financial means [LLMS13].

Table 4.1: Categorization of privacy definitions

The definitions of privacy have evolved. This has prompted the need to take privacy into consideration as a quality for system design. Therefore, tradeoffs have to be found between privacy and other qualities.

For example, complete privacy would mean, roughly, sharing no personal information with anyone. Such behavior would exclude a person from social interactions with others and the environment. Typically, this is not the behavior that is expected from systems such as AmI and socio technical systems, where social interaction plays a crucial role.

Conversely, if a person was to share everything about his/her personal life with everyone, s/he may be considered very social, but risk his/her safety (physical, financial etc), as any mischievous person may use this information in harmful ways. Furthermore, in AmI, a privacy policy with extremely high constraints would not allow any personalization of the system and therefore users would not benefit from its full potential.

4.2.2 Privacy issues

In this section, we discuss the general privacy issues encountered when sharing private data.

4.2.2.1 Identity Disclosure

Identity disclosure occurs when data or aggregate data in an anonymized dataset still allow to identify a unique user. A common practice for protecting identity disclosure is to remove identity related attributes from released data. However, Sweeney et al. [Swe02a] demonstrated that this is not adequate in protecting personal identities. In fact, the author showed that 87% of the population in the United States can be uniquely identified using three demographic attributes: gender, date of birth, and 5-digit zip code. These attributes are normally not considered identity attributes. However, since they can potentially be used to uniquely identify a record, they are called quasi-identifier (QI).

4.2.2.2 Location Disclosure

Geolocation can be used to breach the location privacy of a person. It is usually a problem, when past locations of a person are stored. As illustrated in [BS03], people often do not care if someone finds out where they were a week ago at a specific time, but if someone could inspect the history of all their past movements, then they might start to see things differently. However, this is not always the case, since, even a single record of someone's location at a specific time can cause privacy concerns. For example, if someone is spotted in a cancer clinic, or in the office of anonymous alcoholics, or in a police department, then privacy could also be breached, since logical (probabilistic) assumptions could be made about this person.

4.2.2.3 Inventorying

Inventorying is the privacy problem of identifying the belongings of a person. When an RFID tag has information about the manufacturer, or the cardholder, then the bearers of this tag are subject to inventorying. This means that an adversary could know, for example, the contents of one's bag, the amount of money he carries, the type of medication he carries, and therefore what illness he may suffer from, where he shops, his accessory preferences etc. In the electric load monitoring domain, inventorying can happen by looking for the electrical signature of household appliances. From a very detailed electricity consumption profile, an adversary can infer the list of appliances present at home.

4.2.2.4 Sensitive information disclosure

Lately, there has been a significant increase in the digital medical data being recorded by healthcare organizations. "While the healthcare industry has benefited from information sharing, patients are increasingly concerned about invasion of their privacy by these practices. These growing concerns on privacy led to the Health Insurance Portability and Accountability Act (HIPAA) in 2001 and have increased compliance requirements for health-care organizations" [LM09].

Another interesting source of privacy breach is provided by people who are authorized to access patient data. "Recent studies have revealed that numerous policy violations occur in the real world as employees access records of celebrities, family members, and neighbors motivated by general curiosity, financial gain, child custody lawsuits and other considerations" [BCDS11].

4.2.2.5 Profiling

Profiling activity for the purposes of AmI is a continuous background activity; it includes extracting useful information (like user location, behaviour, room temperature), "enabling the identification of the user's needs, selecting suitable services and adjusting the parameters of the selected services in order to allow the AmI environment to behave according to the users' preferences, actions and expectations" [SHGW05]. Consequently, profiling activity is essential to meet user needs and preferences. The aim of AmI is to anticipate inferred habits and desires, which means that "one does not provide a profile on the basis of what one thinks to be one's preferences, but that one trusts the system to infer them and to adjust the environment accordingly" [HM06].

However, profiling can cause several privacy problems. For example, in the energy consumption domain. Smart meters have unintended consequences for customer privacy [MM09]. Energy usage information stored at the meter and distributed thereafter acts as an information-rich side channel, exposing customer habits and behaviors.

4.2.2.6 Personal Data Matching

Entity resolution [Chr12] is the process of identifying and merging references corresponding to the same real world entity. An *entity* could be a specific person, place, building, etc. Multiple references to a person, scattered through the Web, could be merged to form a single file, containing all the information that has been recorded about this person. This could also affect the privacy of the people that are somehow connected to this person.

Assuming that two databases are linked, one containing medical records and the other containing social information, it could be deduced for example that the citizens of a specific area, or race have higher chances of having a contagious disease. If such knowledge is publicized, this could lead to a racist behavior against this group of people and diminish their chances of employment, or getting certain types of insurances [Chr12]. Various real-world stories related to privacy and data matching have also been described by Clifton et al. [CKD+04] and by Frienberg [Fie06].

4.2.2.7 Purpose Control

Another aspect of privacy breach is discussed in [PPZ11]. Even if only authorized people access personal data, it is still not guaranteed that the personal data will be used for the intended purpose. Preventive mechanisms are not able to prevent a user to process data for

other purposes after the same user has legitimately got access to them. [KS02] identifies the need of three additional elements, i.e. purpose, condition and obligation, besides the basic authorization elements, i.e. subject, object and action.

4.2.3 Privacy preservation techniques

In this section we review typical approaches used to preserve privacy. In general, Langheinrich provides different principles and guidelines for designing privacy-aware systems [Lan01]. A summary of these guidelines, which are often called privacy by design guidelines, is presented here:

1. Notice: When collecting data, it is very important to notify users about it. For example, it is important to announce to users that they are entering an area with video surveillance.

2. Choice and consent: In some situations, it is not enough anymore to simply announce and declare data collection - it is also required that collectors receive explicit consent from the data subject (by written contract, digital signature, check box.).

3. Anonymity and pseudonymity: Collected data cannot be traced back to an individual, or at least unlinkable.

4. Proximity and locality: In essence, information should not be disseminated indefinitely. For example, information collected in a building would stay within the building's network. Anybody interested in this information would need to be actually physically present in order to query it.

5. Adequate security: Use of the best practice in securing communication, storage, etc.

6. Access and Recourse: Trusting a system requires a set of regulations that separate acceptable from unacceptable behavior with a reasonable mechanism for detecting violations and enforcing the penalties.

Privacy issues occur when someone's personal data becomes available, against this person's will. However, there is no issue at all when the same personal data is available, but without the possibility, or, to be more realistic, with a very small chance of connecting them to this person. Fung's et al. survey [FWCY10] provides a typical scenario for data collection and publishing (described in figure 4.1). In the data collection phase, the data publisher collects data from record owners. In the data publishing phase, the data publisher releases the collected data to a data miner or to the public, called the data recipient, who will then conduct data mining on the published data. This technique requires a trusted centralized third party that uses all the collected data from all the users, in order to anonymize well. We present next the most popular and widely known approaches in the anonymization domain.

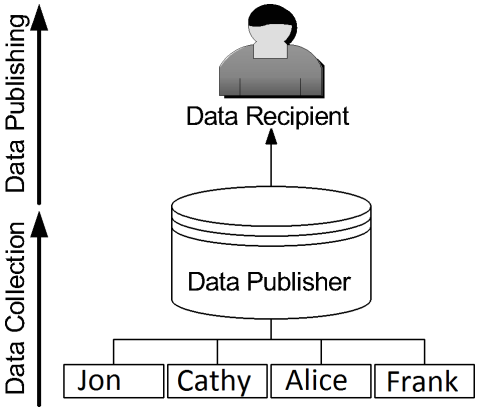

Figure 4.1: Anonymization happens before the data publishing phase.

4.2.3.1 k-anonymity

A very popular approach during the last decade to anonymize personal data before releasing the collected data, has been based on the notion of k-anonymity [Swe02b]. In a k-anonymized dataset, each record is indistinguishable from at least k - 1 other records with respect to certain identifying attributes [XT06b]. k-anonymity can be achieved by suppressing and generalizing the attributes of users in the data [Swe02a]. Suppressing an attribute value means deleting it from the perturbed data and replacing it with a wildcard value that matches any possible attribute value.

Generalizing an attribute means replacing it with a less specific but semantically consistent value. One can see suppression as a special case of generalization, and that suppressing all attributes would guarantee k-anonymity. This is why a notion of utility in the data has to be incorporated whenever sanitizing data. Because the utility and privacy of data are intrinsically connected, no regulation can increase data privacy without also decreasing data utility [Ohm09]. The actual objective is to maximize utility by minimizing the amount of generalization and suppression [BA05a]. Achieving k-anonymity by generalization with this objective as a constraint is a Non-deterministic Polynomial-time hard (NP-hard) problem [PS07].

4.2.3.2 ℓ-diversity

However, a more recent work [MKGV07], introducing ℓ-diversity, has proven that k-anonymity does not guarantee privacy against attackers using background knowledge, or when the sensitive data are lacking diversity.

Consider the example, in which a hospital publishes its 4-anonymized list of patients daily. Bob, suffering from cancer, is one of them, indistinguishable from at least three other patients in the daily list, since they all live in the same neighborhood and they all belong to the same age-group, 60-80. They are the only ones in the list who live in this neighborhood and by chance, they all have cancer. Alice, Bob's neighbor, saw an ambulance taking Bob to the hospital this morning. Alice can easily infer from the hospital's list that Bob has

cancer, since she knows that he is 68 years old, he lives in this neighborhood and he was taken to the hospital that day, so he is one of the patients in that list. So, a list is ℓ-diverse if it contains at least ℓ "well-represented" values for the sensitive attribute S. Anatomy [XT06a] is an example of a linear-time algorithm to compute tables that obey ℓ-diversity requirement.

4.2.3.3 t-closeness

But again, another more recent study [LLV07], shows that ℓ-diversity may be difficult and unnecessary to achieve or insufficient to prevent attribute disclosure. For instance, suppose we are studying the test result for a particular virus (positive and negative). Further suppose that 99% of the results are negative, and only 1% being positive (have the virus). Then the two values have very different degrees of sensitivity. One would not mind being known to be tested negative, because then one is the same as 99% of the population, but one would not want to be known/considered to be tested positive. In this case, 2-diversity is unnecessary for an equivalence class that contains only records that are negative and on the other hand, might be difficult to achieve to cover someone being positive.

Another problem with the ℓ-diversity is that it does not take into account the semantical closeness of the sensitive values in each list. To tackle these issues, the notion of t-closeness was introduced in [LLV07]. The idea behind it is to distribute the records of the table in lists where the distance between the distribution of a sensitive attribute in each list and the distribution of this attribute in the whole table are not higher than a threshold t. This way, the statistics of each anonymized-list regarding the sensitive attribute are "close" to the statistics of the full population. t-closeness is designed to reduce the information gain of an observer, which is the difference between the posterior belief (information acquired after seeing the released lists) and the prior belief (information before seeing the released lists).

4.2.3.4 Differential privacy

Dwork [Dwo06, Dwo11, Dwo08] introduces, and describes a mechanism achieving the notion of differential privacy. It proves that the formalization of Dalenius' [Dal77] desideratum for statistical databases, that "nothing about an individual should be learnable from the database that cannot be learned without access to the database" cannot be achieved. Moreover, it is shown that even the privacy of someone not in the database can be at risk. Differential privacy, intuitively, is the additional privacy risk of someone's participation in a database.

4.2.4 Discussion

The previously cited solutions suits well the data collection and publishing domain. They are mostly made to protect the anonymity of a user inside a group. These approaches protect single user data by averaging or aggregating information with other users in groups

of a minimum size k. For example, when a hospital wants to share patients data to an external third party research lab, they anonymize the dataset so they don't expose their patients to privacy risks.

In the particular case of AmI, these solutions need to be adapted to the fact that hiding the identity of the user is not really the main privacy issue, and to the fact that data from other users might not be available in order to be fed to the anynomization techniques. When an application shares data to a third party in order to get a service, profiling user's habit and behavior becomes the main privacy concern. In a fair trade, we would like that the utility received from the third party to match the quality of data we sent to it. For example, let's say a third party is offering a heating management system for smart homes. Sending very fine grained temperature and electricity consumption measurements every minute to this third party can increase the profiling privacy risk for the user. The third party can learn a lot about the user habits in this case, in return the user could have received the same service with less data sent. Let's say updating the heating system within 10 minutes is enough for the user. For this problem, we propose in chapter 9, **blurring components** to be used as our main privacy preservation elements. We present these ideas in chapter 9.

4.3 Utility

The utility quality reflects the usefulness of the data shared to the third party after being anonymized. In the data collection and publishing domain, there are two approaches to measure utility. In the first approach, one measures the amount of utility that is remained in the anonymized data. This includes measures such as the average size of the equivalence classes [MKGV07] and the discernibility metric [BA05b]. In the second approach, one measures the loss of utility due to data anonymization. This is measured by comparing the anonymized data with the original data. This includes measures such as the number of generalization steps and the KL-divergence between the reconstructed distribution and the true distribution for all possible quasi-identifier values [KG06].

In a first thought, one can perceive that the **utility** measure of the shared data is directly in a linear correlation with the privacy or the anonymization level. Although this can be the case for some situations, it is not the general case for AmI. The privacy measurement reflects how much private information the user is sending to the third party while the utility measurement should reflect how much benefits, the user is receiving in return. For example, when the quality of the original data is low, let's say because the installed sensors are not at all accurate, it should be expected that the utility of the data is low even if no anonymization is done. Similarly, the fact that the anonymized data can be used for a variety of data mining tasks does not imply that the anonymization method is effective. Another anonymization method for a different task may provide higher data utility with the same privacy loss [LL09]. Figure 4.2 shows an example of the relationship between the utility and privacy loss measures in the data publishing domain using the different anonymization methods we presented.

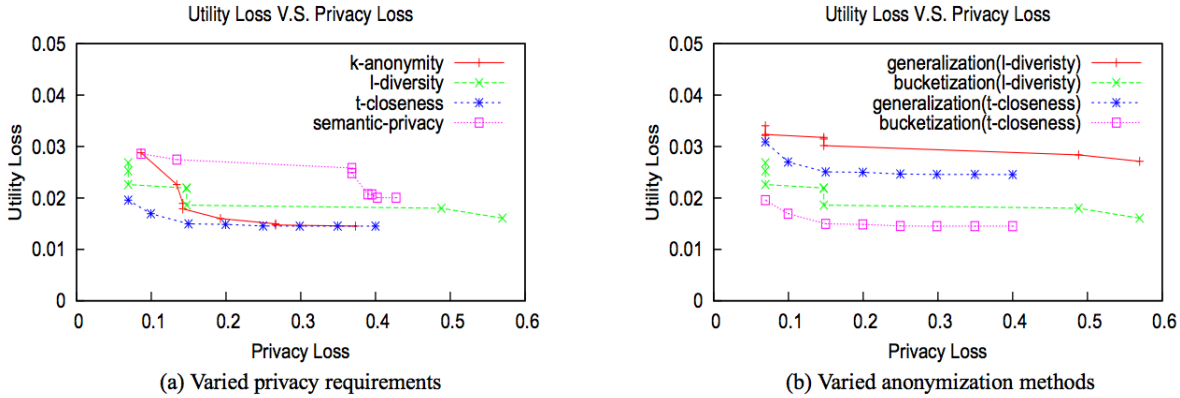

Figure 4.2: Trade-offs between utility and privacy in data publishing.

In figure 4.2, the utility-privacy tradeoff region is defined by the area where a change in one axis leads to a change in the other. This tradeoff region depends on the anonymization technique and on the dataset. Ideally, we would like to find solutions in this region.

4.4 Efficiency

Although we are mainly interested in finding a good tradeoff between privacy and utility according to the context, we should not forget the constraint that AmI systems should be reactive in near real-time, and the fact that devices have limited capability. In this section, we will present the potential efficiency limitations that can be taken into account when searching for a privacy-utility tradeoff.

1. Computation capability: anonymizing data streams can be intensive in term of CPU power. If two anonymization techniques offer the same privacy and utility level, we would prefer to select the one which is less computation intensive. As a metric in this domain we can use the average execution time or number of CPU cycles for the anonymization routine.

2. Memory usage: in small devices, we would like to favor anonymization techniques with small memory fingerprint over the techniques with higher memory fingerprint.

3. Network usage: when sending data over the network is costly, like in the case of smartphones over a 3G connection, the amount of data sent on the network per time unit can be used as an efficiency metric as well.

4. Energy consumption: If the mobile device is running on battery, energy efficiency in term of watts consumed can be used as well as an efficiency metric.

The efficiency quality is orthogonal to privacy and utility and need to be taken into account when searching for a tradeoff for AmI.

5

Reasoning tools and techniques

In this chapter, we present the related work on reasoning tools and techniques, mainly the multi-objectives optimizations algorithms and the state of the art of machine learning. The MOEA will be used as main tool to find the trade-offs between the different qualities presented earlier. The machine learning background is important to understand how we will enable context awareness automatically with the least user intervention.

Contents

5.1 Introduction

In order to achieve the smart aspect in ambient intelligent applications, the system needs to have reasoning tools that optimize itself for the current context, and auto-discover correlations and hypothesis on the system usage. In this chapter, we introduce first, the background related to optimization problems and algorithms in general, then we present evolutionary algorithms and multi-objective evolutionary algorithms in particular. These algorithms will be used to find a good trade-off between the different qualities in the AmI system according to the context. Then we introduced the different categories in the field of machine learning, and the different reasoning tools and algorithms available. These algorithms, will contribute to the automatic context discovery rules, and the automatic profiling of qualities required for every context. In a first step, we considered that the context rules and the qualities to optimize per context are defined manually by an AmI expert or by the AmI user. Our long term goal, is to include smart monitoring of the data by mean of machine learning. This will allow AmI systems to automatically infer these rules, without the need to ask them from the user of system designers. Machine learning techniques will allow us to respond to the unobtrusiveness challenge in AmI.

5.2 Optimization problems

Optimization problems fall into two categories: problems that can be solved with an exact technique in a reasonable amount of time, and the others, which require the introduction of metaheuristics to find near-optimal solutions. Metaheuristics are used when the solution space is extremely large, making it practically impossible to evaluate all the solutions to find the best one, and where there is no known efficient technique to find the exact best solution. The general idea is, in such cases, to find approximate solutions by defining a way to decide which potential solution is the best among several. Potential solutions are in this case compared in terms of **fitness**, noted f, and the objective is to find the best solutions according to that fitness.

5.2.1 Solution encoding

The first step to do in optimization problem is to select an appropriate solution encoding for the problem to optimize. Different encodings have been used in literature [CVVL02], *e.g.* binary, permutation, matrix, graph, and tree based encodings. Many proposals have been presented for which encoding is suitable for which type of problem [CVVL02]. For example, graph-based encodings are suitable for network optimization problems. Some encoding types, like permutation-based encodings, have the advantage to require only a very small memory footprint. Others, such as matrix encoding, can use optimized mathematical libraries to perform fast algebraic operations as operators. The aim of the encoding step

is to extract the variables that are subject of optimization in order to define the solution space that the optimization algorithms will work on.

5.3 Evolutionary algorithms

In the 1960s, several researchers independently suggested to adopt the principles of natural evolution (Darwin's theory) for optimizations. This created the field of evolutionary algorithm (EA) which form search-based heuristics mimicking the natural evolution process. They represent a smart way to randomly search for solutions to optimization problems. To apply such an approach, several parameters have to be defined. In EA, a solution vector is called *individual* or *chromosome*, which consist of discrete units, which are called *genes*. Each gene controls one or more features of a solution. Usually, an individual corresponds to a unique solution in the solution space. This requires a mapping between the domain specific solution space and chromosomes which is called *genetic encoding*. EAs operate on a collection of individuals, called a *population*. The population is usually randomly initialized and then evaluated with a provided *fitness function* in order to select the *"most appropriate"* one for the next *generation*. The fitness function quantifies the individuals' ability to solve the optimization problem. After this step, EAs use three operators to generate new solutions: selection, crossover and mutation [HM10]:

1. **Selection** chooses individuals for performing crossover and mutation. The selection is made by choosing the individuals with the best scores according to the fitness function.
2. **Crossover** selects two individuals and switches some of their genes. This is usually performed by ordering the individuals' genes and switching all the genes after a randomly selected point. Crossover results into two new individuals called offsprings.
3. **Mutation** performs on an offspring by changing the values of one or more of its genes.

Performing the selection, crossover and mutation operations on a population results in one evolution cycle of the population. This cycle is called population generation. At each generation, the individuals that fit the best to the problem, i.e., which have the best fitness, are kept. This principle is adapted from the evolution's theory, i.e., survival of the fittest. An overview of one generation is presented in figure 5.1. The algorithm terminates after completing a predefined number of generations or after the fitness function attains a certain threshold level.

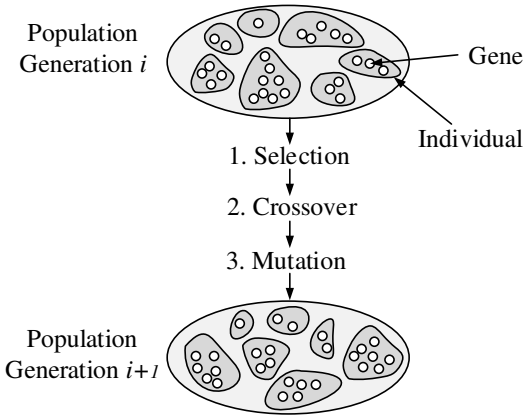

Figure 5.1: The process of evolving a population in evolutionary algorithms. The current population is evolved into a new one by selecting, crossing and mutating individuals.

5.4 Multi-objective optimization

Multi-objective optimization (MOO) refers to the process of optimizing more than one objective at the same time. The aim of these approaches is to search for optimal (or nearly optimal) solutions requiring trade-offs between two or more conflicting objectives.

Let X be the set of all the possible solutions to a problem and let $\mathbf{F} = [F_1(x), ..., F_k(x)]^T$ be a vector of k objective functions. If each objective has to be minimized, MOO aims at finding $x_1, ..., x_k$, i.e., the solutions to the problem, such as $\mathbf{F}$ is minimized. The minimization of $\mathbf{F}$ is the process of optimizing simultaneously the k objective functions [MA04]. The terms objective function and fitness function are similar, but generally fitness is used for single-objective optimization while objective is used for MOO problems.

5.4.1 Pareto front

Definition 3 *"The Pareto optimal set is a set of solutions that are non-dominated with respect to each other. While moving from one Pareto solution to another, there is always a certain amount of sacrifice in one objective(s) to achieve a certain amount of gain in the other(s)" [KCS06].*

Let x_1 and x_2 be two potential solutions to a MOO problem. We say that x_1 dominates x_2, written as $x_1 \succ x_2$ if and only if $\forall i \in \{1, ..., k\}\, F_i(x_1) \leq F_i(x_2)$ and $\exists i \in \{1, ..., k\}\, F_i(x_1) < F_i(x_2)$. Given $x_1, ..., x_n$ potential solutions to the MOO problem, the **Pareto front (PF)** corresponds to the subset of these potential solutions that are non-dominated by the others.

An example of PF is illustrated by figure 5.2 for two objectives F_1 and F_2 to minimize. In this example, x_1, x_2, x_4, x_6 and x_7 are in the PF set since they are not dominated by any

other solution. By contrast, x_{10} is dominated (among others) by x_2 ($x_2 \succ x_{10}$). So, it does not lie on the front. Finally, we denote as PF size the number of solutions in the PF.

Finally, multi-objective evolutionary algorithms denotes a set of evolutionary techniques used to solve MOO problems. The first MOEA was proposed by Shaffer [Sch85].

There are several algorithms to select which solution subsets of the Pareto optimal set to keep in order to cover, as diversely as possible, the different trade-offs between objectives. The NSGA [DPAM02] algorithm family (*e.g.* NSGA-II, NSGA-III) is amongst the most well known ones. MOEAs, like conventional EAs, rely on a genetic encoding step. However, besides mutation and crossover operators, for MOEAs several fitness functions representing the different objectives, need to be defined, instead of only one fitness function in EAs.

5.4.2 Selecting a solution

At the end of MOEA execution, a pareto front offering several tradeoffs between the different objectives is presented. A decision making procedure has to be called to select one solution from the final Pareto subset of solutions. If the user privilege one objective over the others, the solution offering the highest score in this objective from the pareto front is selected. In other situations, flattening the several objectives using any linear or non-linear combination can be a way to select the final solution to deploy. In this domain one of the most famous method is the so called "elbow-method", in other references it is known as the "knee-method" [BDDO04]. The idea is to select the solution that is on the elbow of the pareto front after normalization of the objectives in an interval from 0 to 1. The rational behind this technique, is that it offers an ultimate trade-off between the different objectives. Meaning that, optimizing any objective furthermore will be at the cost of deteriorating the other objectives a lot. Visually speaking, if we draw the pareto front of solutions in a optimization problem consisting of two optimization functions f1 and f2, this technique aims to find the point in the middle of the red zone in figure 5.3.

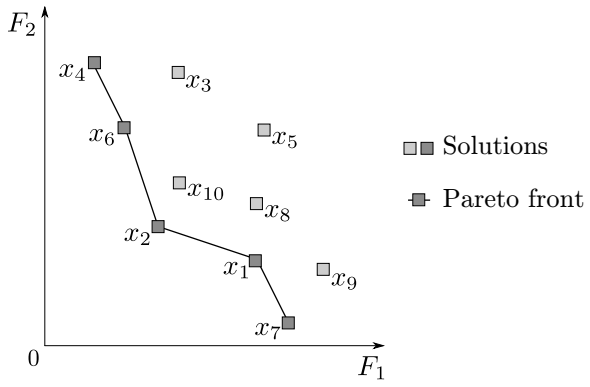

Figure 5.2: An example of Pareto front with two objectives F_1 and F_2 to minimize. The solutions x_1, x_2, x_4, x_6 and x_7 are in the Pareto front since they are not dominated by any other solution.

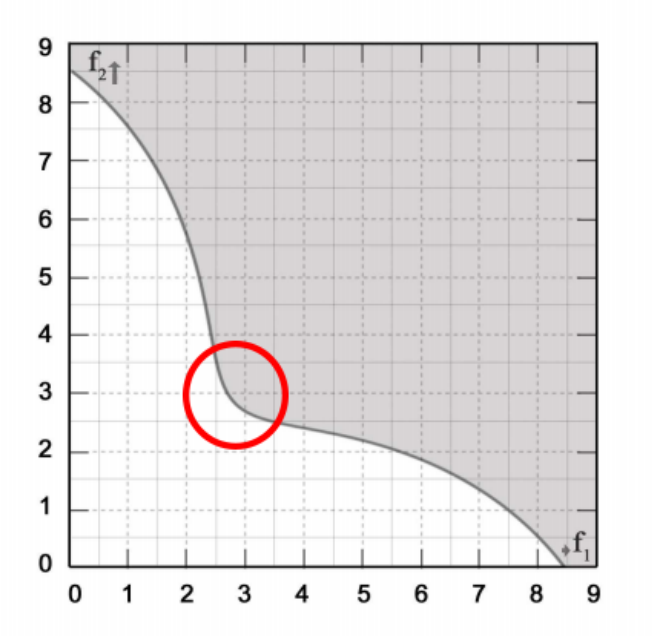

Figure 5.3: The knee in the pareto front, when optimizing (minimizing) two fitness functions.

5.4.3 Discussion

Although MOEA are very famous and well deployed, several challenges still exist in order to adapt them for the AmI domain. First of all, and most importantly, none of the current encodings is suitable for complex domains containing different types of data, attributes and relationships, of variable non-predefined size, at the same time. Therefore, it is necessary to have a complex encoding/decoding step to transform the domain specific multi-objective optimization problem into a suitable MOEA representation and to transform the solution back to the domain problem. The encoding step is complex and can be sometimes even more complicated than the actual optimization problem itself [KCS06]. In this thesis, we discard the encoding step by proposing to do the optimization directly on the domain specific model itself, making the domain specific model its own MOEA encoding. This approach is presented in more details in chapter 8, and applied for the AmI domain in chap 9. Secondly, the elbow method might not be suitable all the times [BDDO04]. For instance, in some problems several elbows can be present, or even none (for example if two fitness functions are linearly correlated, the pareto front is a straight line and no elbow is present). Moreover in AmI, we might have a-priori for each context, before running the optimization, an idea of which region of the trade-off reply the best for the need of the current context. Thus asking MOEA to provide a full pareto front offering all the whole spectrum of all possible trade-offs on the different objectives wastes resources.

5.5 Machine learning

Machine learning is an evolution of pattern recognition and computational learning theory in artificial intelligence. Machine learning explores the construction and study of algorithms that can learn from and make predictions on data. It uses algorithms operating by building a mathematical model from example inputs in order to make data-driven predictions or decisions, rather than strictly static program instructions.

Machine learning is strongly tied to computational statistics, it is also used to make predictions, simulations and it is tied to mathematical optimization. When employed in industrial contexts, machine learning methods may be referred as predictive analytics or modeling.

The essence of Machine learning is to create compact mathematical models that represent abstract domain notions of profiles, tastes, correlations, and patterns that 1) *fit well* the current observations of the domain and 2) are able to *extrapolate well* to new observations. These two objectives define the two main problems in most of machine learning subfields which are:

1. **Under-fitting**: where the machine learning algorithm was not able to learn well from the current observations and has poor accuracy and performance even on the already well known samples.

2. **Over-fitting**: where the machine learning algorithm fits too well the training data with very good performance and accuracy to a certain point that it looses its ability to generalize on new data (outside of the training set).

5.5.1 Categorizations of ML

Several possible categorizations of machine learning techniques are possible. We can divide the techniques according to the types of problems they are trying to solve, or according to the type of learning that is used, or according to the frequency of execution of the learning algorithm. In this section, we present these different possible categorizations.

5.5.1.1 Type of problems

The first categorization is according to the type of problems they are trying to solve.

1. **Hypothesis based**: In these problems, we can suppose that there exist a best mathematical function (or hypothesis) that links the expected output to the input. For example a linear regression hypothesis, like in the case of evaluating the price of the house from several of its features: we can say that the price is a linear combination of the other attributes of the house. The learning consists then on

fitting the multiplicative weight parameters of the different attributes to the different presented training samples.

2. **Density estimation**: The problem here is to learn a probability distribution of inputs over a certain domain space. This can create a simulator that generates data with similar statistical properties of the domain space. (For example learning the electricity power consumption profile by Gaussian kernels, then we can consider that the learned distribution will be the same the next day).

3. **Classification**: It is the problem of identifying to which of a set of categories (sub-populations) an observation data belongs to, on the basis of a training set of data containing observations with a previously known category membership.

4. **Clustering**: It is a similar problem as the classification, but the different classes or categories are not known a-priori.

5. **Anomaly detection**: It is similar to classification, but these problems usually have only two classes (normal and abnormal) and there is a big bias in the distribution of items in the classes (much more items in the normal class and few in the abnormal).

6. **Dimensionality reduction**: It is the problem that, from a set of input attributes, select the most appropriate ones relevant for a specific learning problem. Dimensionality Reduction techniques can be useful to simplify input data (by removing redundant attributes) which can accelerate a supervised learning method afterward.

7. **Recommendation system**: The aim is to provide from a set of user,product,rating, recommendations for users of potential products they might like. Some recommendation systems are composite from Linear regression and clustering algorithms. For example: cluster the users or products to similar user/products groups, than the recommendations are generated from the weighted average from the user/product neighbors.

5.5.1.2 Type of learning

The second categorization is according to the type of machine learning algorithms used to solve the previously defined problems.

- **Supervised**: In supervised learning, the data has predefined and well-known fields to serve as expected output of the learning process such as a classification category (for ex.: spam, not spam) or to estimate the value of a domain specific field such as the price of houses. In this type of learning, a mathematical model is prepared through a training process where it is provided by input features and their corresponding expected outputs. The learning consists of optimizing the mathematical model, so the provided input features map well to the provided expected output. The training process continues until the mathematical model achieves a desired level of accuracy on the training data between the inputs and outputs. Some example problems of supervised learning are classification and correlation problems. For instance: Learning

the price of a house from several of its features (gps position, area, number of rooms, year of construction...). The training set consists of several examples of houses defined by their features and their prices. The ML creates mathematical functions, for example a linear regression of the attributes. The training process is to find the best parameters of this regression that can fit well the data. In this sense, supervised learning can be seen as a function approximation between output and input attributes.

- **Unsupervised**: For unsupervised learning, the input data is not labeled and does not have a known field defined as output. The machine learning algorithms try to deduce structures present in the input data. Unsupervised learning can find hidden patterns in the input data. An example problem is clustering: a list of users and their preferences is provided and the algorithm has to divide them into groups or clusters of users of similar taste. In this sense, unsupervised learning can be seen as a concise compact and compressed description of the input.
- **Semi-Supervised**: In this case, the input data is a mixture of labeled and unlabeled examples. There is a desired prediction problem expected but at the same time, the model must learn the structures to organize the data as well as to make predictions.
- **Reinforcement learning**: In this type of learning, the output data is provided as stimulus to the machine learning model from the environment to which the ML model must respond and react by taking some actions. The feedback is provided in terms of punishments and rewards from the environment. The reinforcement learning algorithms attempt to find a policy that maps states of the world to best actions to be taken in those states in order to maximize the expected reward from the environment. Reinforcement learning differs from the supervised learning problems in the fact that correct input/output pairs are never presented explicitly and directly. In this sense, reinforcement learning can be seen as supervised learning but from a delayed reward function. At the same time, the difference with the unsupervised learning is that in reinforcement learning there exists somehow a kind of reward function that guides the learning, where in the unsupervised learning, no expected output or feedback is presented.
- **Meta-Learning**: It is the process of learning the parameters of a learning algorithm itself to ensure that both under-fitting and over-fitting problems are solved. Usually it involves dividing the current available data into 2 sets (training and testing sets), then training the learning algorithm on the first set and testing on the second to get two performance measures on how well the algorithm fits the data and how well it can generalize to new data it was not trained on. The meta-learning optimizes the learning parameters of the learning algorithms until getting good results in both directions.

5.5.1.3 Frequency of learning

The third categorization is according to how often and when we are executing the machine learning algorithms.

1. **Full Online learning**: For every new observation of Input/output (supervised) or just input (unsupervised), the learning algorithm is executed, and its state is updated, then the training vector is discarded. The advantage of this type of machine learning is that the learning is atomic, incremental and always evolving according to new observations. On the other hand, the disadvantage is that it might have lower accuracy than full-offline learning algorithms for small training sets.

2. **Lazy learning**: Whenever there is a need to estimate an output for a specific input vector, the learning algorithm is trained by a small batch of observations close or similar to the one needed (for example the nearest K neighbors similar to the requested input). This learning type offers a case-based or context-based reasoning meaning that the learning is tailored to the requested specific input. Moreover, it is on-demand, meaning that no processing will be lost on training for all possible contexts, or inputs, as long as they are not requested for the prediction. However the disadvantage of this type is that it is slower that the full online (becasue we need to resolve, fetch and train for K nearest vectors before being able to predict), thus it will take more delay to respond. Plus, it requires a clustering function that provides the nearest neighbors to learn from whenever an input is presented.

3. **Batch or offline learning**: This is a very slow and time consuming process. The learning is fully offline and works usually on the whole dataset of training examples. However, once trained, the learned model is used to estimate values till the next learning update phase (where everything needs to be reconstructed and retrained from scratch). The advantage of this type is that usually it achieves the highest accuracy, however this accuracy gets worst with time till the next learning update phase. The risk here is that the learned model can be outdated if the domain evolves quickly between two update phases. Another disadvantage is that usually this type of learning is very consuming in term of computational resources (Cpu and memory)

5.5.2 Machine learning for the AmI domain

In an AmI environment, learning means that the environment has to gain knowledge about the preferences, needs and habits of the user in order to better assist the user [GPR06]. Furthermore, it is insufficient to learn user patterns only once because preferences and routines can change with time. Hence, it is necessary for an AmI environment to adapt itself to these new patterns continuously [RJD05]. Learning and adapting to user patterns is an essential feature of an AmI system. In a smart home, for example, it allows the system:

- To detect and classify current context.
- To understand common behaviors of the user (profiling).
- To automate activation/deactivation of devices depending on the needs of the user.
- To make the system more efficient (e.g. turning the lights off when the user leaves home).

- To increase safety (e.g., by either switching the cooker off or issuing an alarm when detecting that the user has left it on).

On the other hand, not all machine learning techniques and algorithms are suitable for the AmI domain. For example, in real-world environments large sequences of data may not be available in advance, may take too much time to gather, or they can be very expensive in terms of computation power to process [KSL10]. For a reactive system, operating in near real-time is a crucial requirement. In this thesis, we focus on machine learning techniques that are best suited for the AmI domain, we are interested in ML algorithms with the following characteristics:

1. The algorithms should be able to create or update the models whenever new data arrives (on the fly).
2. The computational effort needed for a single update should be minimal and should not depend on the amount of data observed so far.
3. The update should only depend on the latest observed value and should not explicitly require access to old data.
4. The generated models should be compact and should not grow significantly with the number of observed instances.

For these reasons, we work on how to model machine learning techniques and integrate them in models@run.time. We focus on the online learning for its suitability for our AmI problems. These ML tools will be useful for automatic context rules detection. At the same time, we show how contextual information can be used to improve domain specific machine learning. This will be discussed in more details in chapter 10.

Part II

Contributions

6

A Continuous and Efficient Model for IoT

Internet of Things applications analyze our past habits through sensor measures to anticipate future trends. To yield accurate predictions, intelligent systems not only rely on single numerical values, but also on structured models aggregated from different sensors. Computation theory, based on the discretization of observable data into timed events, can easily lead to millions of values. Time series and similar database structures can efficiently index the mere data, but quickly reach computation and storage limits when it comes to structuring and processing IoT data. As a first contribution, we propose a concept of continuous models that can handle high-volatile IoT data by defining a new type of meta attribute, which represents the continuous nature of IoT data. On top of traditional discrete object-oriented modeling APIs, we enable models to represent very large sequences of sensor values by using mathematical polynomials. We show on various IoT datasets that this significantly improves storage and reasoning efficiency.

This chapter is based on the work that has been published in the following paper:

- Assaad Moawad, Thomas Hartmann, François Fouquet, Grégory Nain, Jacques Klein, and Yves Le Traon. Beyond discrete modeling: A continuous and efficient model for iot. In *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 90–99. Conference Publishing Consulting, 2015

Contents

6.1 Introduction

The Internet of Things (IoT) and the omnipresence of sensors is considered to become one of the next disruptive changes in our everyday lives [Gar14]. Intelligent systems will be able to analyze our past habits and to anticipate future trends based on data measured and aggregated from various sensors. Since numerical sensor values alone are usually not enough to reason, intelligent systems often leverage modeling techniques to represent structured data contexts. The models@run.time paradigm has demonstrated its suitability to represent and monitor cyber-physical systems and their contexts [BBF09], [BFCA14]. Data analytics [DD13] and machine learning techniques [LLS+13] can analyze these data contexts, *e.g.,* to understand and forecast our habits by computing user profiles.

Today's computation theory is based on the discretization of observable data into events associated to a finite time [Hog92]. Following this theory, even the eternity (*i.e.,* long term tasks) can be observed and computed as a sequence of finite temporal data points. Each observable event is then stored by augmenting data with a timestamp, for instance by creating a model element instance with an attribute timestamp [Pla]. The discretization process of a sensor can easily lead to millions of values in a short amount of time, *e.g.,* considering domains like industrial automation, automated smart homes, traffic monitoring, monitoring of weather conditions, or medical device monitoring. The raise of IoT leads to a massive amount of data, which must be stored and processed. For this reason it is often associated with Cloud and Big Data technologies [GBMP13] ,[Pla]. Specific data structures like time-series [GBT94] [ore13] can optimize the indexation of mere data. However, the required storage and computation power to store and process this amount of data are a challenge, especially for stream and real-time analyses [HAG+13]. Jacobs [Jac09] even calls the modeling of timed data as enumerations of discrete timed values, *Big Data pathology* or *Big Data trap*. Regardless of the chosen solution, intelligent systems need a structured representation of timed data in order to (periodically) reason about it. Therefore, building this context in a way that reasoning processes can efficiently process it, is critical.

Nevertheless, the physical time is continuous and this data splitting is only due to the clock-based conception of computers. In contrary, IoT related data are collected value by value (due to regularly sampled measurements) but are virtually continuous, meaning that, for example for a temperature sensor, at any point in time it should be possible to extrapolate a value (even if there is no measurement at this specific point in time). Indeed, the absence of a data record for a temperature value at a time t does not, of course, mean that there is no temperature outside at this time t . It could simply be that the sensor is saving energy by reducing its sampling rate or it is just not fast enough to measure the temperature at the requested granularity.

The last measured temperature, recorded for time $t - 1$, would be the closest value to the requested one. This is due to the fact that the physical value in reality is continuous and with this semantic (taking the last measured value), this can be a solution. In addition, measurement inaccuracies are an inherent part of all physical measurements and strongly

considered in signal processing techniques. However, neither the continuous nature of physical measurements nor measurement inaccuracies are considered in modeling techniques. We claim that the traditional mapping between discrete data and model elements is inefficient for the very volatile and continuous nature of IoT related data. In fact, a discrete modeling strategy relies on the enumeration capability of data that can be, in practice, very costly or even impossible, facing the amount of IoT data. Looking at techniques used in signal processing [KCHP01], they offer strategies to represent signals as mathematical functions. Considering the major research objective of this chapter, we formulate the hypothesis that techniques from signal processing can create reasoning models that can bridge the gap between continuous volatile data and complex structures.

Because models should be *simpler, safer and cheaper* [RWLN89] than the reality, in this chapter we introduce a new meta attribute type for models and models@run.time, which can dynamically encode continuous IoT related data. We offer as well an algorithm that enables models@run.time to virtually represent large sequences of sensor values without the need to store all discrete values of the sequences. This keeps the simplicity and expressibility of discrete object-oriented modeling APIs, while using mathematical polynomials to store and read values in an efficient manner. Our approach leverages a live learning strategy, which enables to encode volatile attributes of models and bridges the gap between modeling and signal processing techniques. We integrated our approach into the Kevoree Modeling Framework and evaluated the performance of read, write, and extract operations on IoT datasets from various domains and storage databases.

In this chapter, we reply on how to represent in a continuous and efficient way the data coming from sensors in the AmI agent local knowledge base. This is presented in figure 6.1.

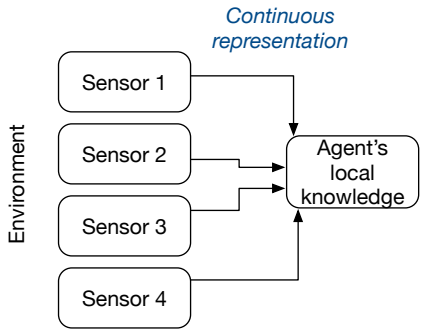

Figure 6.1: Local knowledge of an AmI agent.

6.2 Time Series and Signal Segmentation Algorithms

Time series are one of the oldest [GBT94] structures to represent timed data and even nowadays still one of the most used [Inf14] ones, especially for monitoring metrics. As with

most software engineering problems, data representation is a key for efficient and effective solutions [KCHP01]. The simplest representation of a time series is a collection of discrete records, each containing a timestamp and the measured value. Such representations are still widely used, because they match perfectly with unstructured storage systems like NoSQL, as for example depicted in the Apache Cassandra blog [Pla].

The major advantage of this representation is that it does not rely on any knowledge about the continuous nature of the encoded signal. However it is not a very efficient solution. For instance, a constant temperature in a room would be encoded with a number of redundant values, based on the sensor sampling rate, whereas a single value would be sufficient. Even by considering theoretically infinite storage capabilities, like in Cloud computing, reading a large amount of records from disk would be a bottleneck for applications requesting historical values (*e.g.,* to learn patterns or make predictions).

Several higher-level representations of time series have been proposed, including Fourier Transforms [KCPM01], Wavelets [CF99], and Symbolic Mappings [DLM$^+$98]. Each of them offer a different trade-off between the compression of the representation and the introduced inaccuracy compared to the original signal. In a nutshell, these techniques rely on a segmentation approach which can be classified as follows [KCHP01]:

1. Given a time series T, produce the best representation using only K segments.
2. Given a time series T, produce the best representation such that the maximum error for any segment does not exceed some user-specified threshold, $maxerror$.
3. Given a time series T, produce the best representation such that the combined error of all segments is less than some user-specified threshold, $totalmaxerror$.

Most of these techniques work only in batch mode, thus they require to load and keep all the signal samples in order to compute the mathematical model. This mathematical model can be later used to extrapolate any signal value between the first and the last well-known measures. As argued by Kristan *et al.,* [KL14], batch processing is incompatible with live reasoning over IoT data and thus also with the envisaged models@run.time paradigm. Firstly, because sensor data are pushed in a stream-based way and thus recomputing the whole model would be very expensive (recomputing for every new value). Secondly, batch processing relies on the enumeration capabilities, which has been depicted as difficult, especially for near real-time reasoning on IoT data. Therefore, for the envisaged models@run.time usage, only techniques which can compress in live *i.e.,* compute signal segments without past values, are appropriate. Using live encoding algorithms, once a signal sample has been encoded into a compressed representation, it can be discarded, therefore such techniques allow to go beyond enumeration limits of samples.

Among the three segmentation strategies, the first and third rely on a strong knowledge of the measured signal. However, such a requirement is usually incompatible with IoT and models@run.time usage. Firstly, models@run.time leverage static information defined in meta models, in which it is difficult to express characteristics about dynamic signal behavior. Secondly, the sampling rate in IoT domain varies considerably, because sensors

often slow down sampling rate for energy efficiency reasons. Thus, knowledge about the best segmentation strategy could be outdated over time, *i.e.,* no longer appropriate. Therefore, the second approach of guaranteeing a maximum error per segment, is the most appropriate, because it is typically domain dependent and it could be defined in a meta model. By using the sensor physical measurement error to guide the compression, we are searching for the most efficient representation for sensor time series according to what we measure from the real world.

To sum up the requirements necessary to be integrated into the models@run.time approach, we need a time series compression and segmentation algorithm which is able to compute mathematical models in live and should be driven by a maximum tolerated error.

One of the most famous and frequently used segmentation solutions in the literature is the Piecewise Linear Representation (PLR) [KCHP01]. Intuitively, PLR refers to the approximation of a time series T, of length n, with K straight lines. Because k is typically much smaller than n, this representation makes the storage, transmission, and computation of data more efficient with a compression ratio of n/K. In our work we extended the PLR representation to a polynomial representation, and most importantly, we do the calculations in live (not in batch). We encode the time series into K segments of polynomials of any degrees (instead of lines).

Fitting a polynomial regression of a degree n on a dataset, is a classical mathematical and machine-learning problem. For a given dataset of m pairs $\{x_i, y_i\}$, with $0 \leq i \leq m$, the question is to find the coefficients $\{a_0, ..., a_n\}$ of the polynomial $f(x) = a_0 + a_1x + a_2x^2 + ... + a_nx^n$, that represents the best, with the least possible error, the given dataset. The error minimized is usually the sum of the squares of the differences between the polynomial representation and the actual values y_i in the dataset: $e = \sum_{i=0}^{m}(f(x_i) - y_i)^2$. Many techniques have been proposed and implemented to solve this problem. Most notably, using linear algebra and matrices manipulation, like the EJML linear solver [Abe10]. However these techniques are not adapted for online and live learning, since they require access to past values and that a user specifies the maximum degree of the polynomial. To workaround this problem we leverage a live splitting of polynomials into a sequence of polynomials valid for time boundaries. We argue that this representation is efficient in terms of storage and yet very fast to calculate in an online manner, making it a perfect choice to be integrated in a models@run.time approach.

6.3 A Continuous and Efficient Model for IoT

Models@run.time and *models* in general are defined by the aggregation of a set of model objects. Each *object* contains a set of *attributes* and a set of *relationships*.

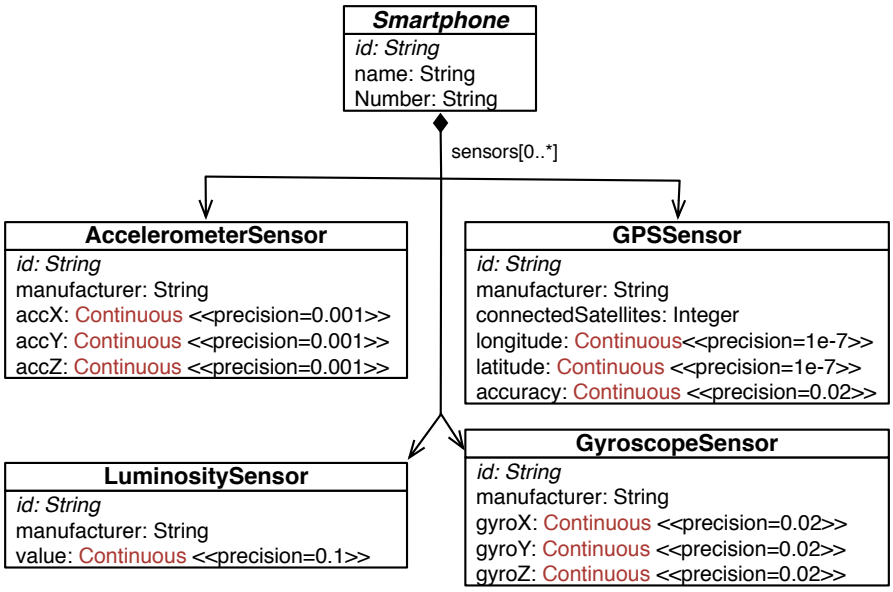

Figure 6.2: A smartphone meta model

6.3.1 Continuous Models Structure

In this chapter, we introduce a new meta-attribute named **continuous**. Figure 6.2 shows the usage of such type in a meta model of a smartphone example. A smartphone has a collection of sensors, each sensor has several attributes. Some of these attributes represent physical continuous signals and they are marked as continuous in the meta model. The precision depicts the maximum tolerated error we can use for our representation and segmentation. Before this contribution, all of these attributes are encoded as discrete doubles, and each minor change of any value of any of these attributes will create a new version of the model. Our hypothesis is that, similarly to time-series compression techniques, a model could detect the relationship between consecutive write on numerical values and try to encode them in an efficient manner. This way, the model will transparently reduce the number of created segments per object while allowing a reasoning engine and the sensor to perform classical get and set operations on any model element attribute.

However, models should be able to contain non continuous attributes such as a string or a reference, any modifications of these will lead to the creation of a new segment as well. For example, in our meta model example, the number of connected satellites in the GPS class is an integer and is not continuous. Each modification on this attribute (aka number of connected satellites change) will create a new segment, regardless if the other attributes (latitude, longitude) can be still encoded in the same continuous polynomial function.

In figure 6.3, we depict this new modeling architecture which is able to manage seamlessly continuous and discrete classical attributes *(e.g. string, boolean...)*. Each object has a balanced tree structure to index the segments it contains. Segments are considered as the smallest data granularity, which can be saved and loaded from scalable storage like NoSQL

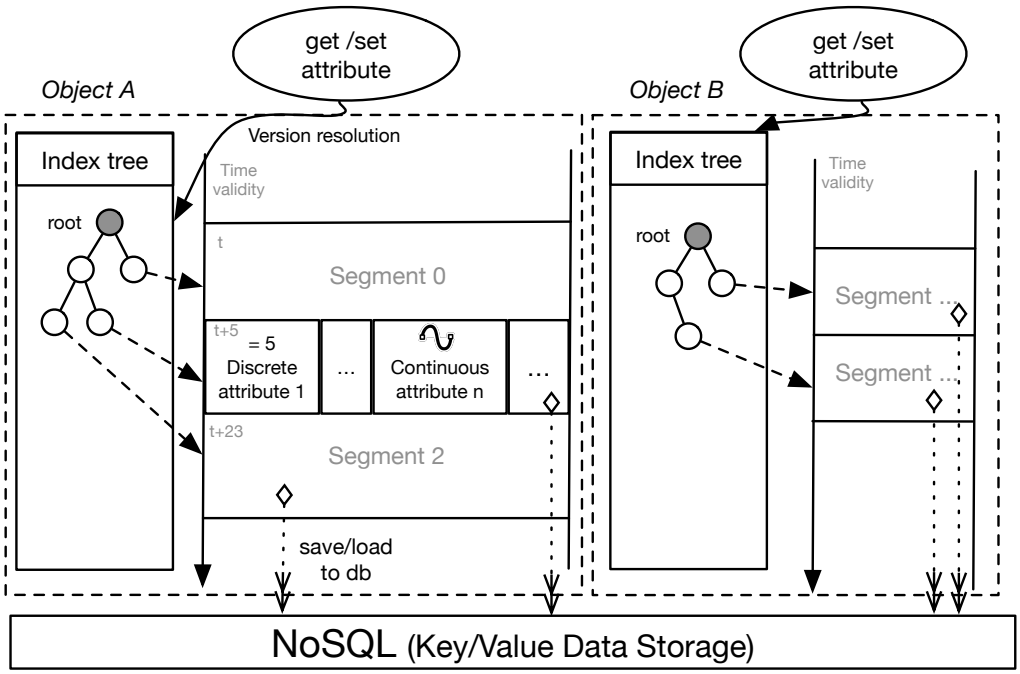

Figure 6.3: Structure for Models with Discrete and Continuous Attributes.

or distributed hash tables technologies. Each segment contains the payload of a version for a model object. This payload contains discrete information like integer values, or functions which can encode several values in a time-series manner of a continuous attribute. Each time a get operation is performed on the model on a continuous attribute, the corresponding segment is loaded and the attribute value is extrapolated using the function stored into the segment. Conversely, each time a set operation is performed on a continuous attribute the model has to decide whether to store it in the existing function (update the function) or whether to create a new one in a new segment for this update. We have defined an online learning algorithm that can build this series of segments, minimize the total number of segments needed while ensuring a maximum tolerated error for the created functions according to the precision of the attributes defined in the meta model, in an online manner (on the fly) suitable for IoT applications. The next subsection details this segmentation strategy, the following one shows the construction of the segments and their continuous functions based on polynomial representations.

6.3.2 Live Model Segmentation Driven by Tolerated Error

A main target of our approach is to produce as few segments as possible under the following design constraints:

- **Computation requirement**: Each segment should be built in a fast online processing way (providing a fast setter) and should be able to compute fast the attribute whenever asked (providing a fast getter).

- **Error requirement**: All continuous attributes in a segment should not deviate from the original data more than the specific tolerated error defined in the meta model.
- **Compression requirement**: The model element will not create a new segment as long as, in its last segment version, the error requirement is met in all the continuous attributes and as long as the other non-continuous attributes did not change. This is to minimize the number of segments created, and to keep the model size minimal.
- **Time validity**: Each segment is valid from its creation time *(time origin)* up till the time origin of the following segment. The time origin of each segment is used as the reference starting point for the attribute functions.
- **Continuity requirement**: The first value of a continuous attribute calculated by one segment should be the same as the last value generated by the previous segment.
- **Self-containment requirement**: Each segment should contain all the information needed for its own process (compression, reconstruction).

Formally, in our current implementation, a **model element** contains a unique ID and an Index tree $tree$.
The **index tree** is a sequence of segments: $tree = \{s_1, \dots, s_i, \dots, s_n\}$.
A **segment** s_i is a 4-uple $s_i =< t_{oi}, C_{Ai}, D_{Ai}, R_i >$, where t_{oi} is a time origin, C_{Ai} is a set of continuous attributes, D_{Ai} is a set of discrete attributes, and R_i is a set of relationships.
A **continuous attribute** $c_{ij} \in C_{Ai}$ is a sequence of weights of the function that represents it. $c_{ij} = \{\dots, w_{ijk}, \dots\}$. In our implementation, we use the polynomial function:
$f_{ij}(t) = w_{ij0} + w_{ij1}(t - t_{oi}) + ... + w_{ijn}(t - t_{oi})^n$,
as the retrieval function to get the value of a continuous attribute at a time t, with $t_{oi} \leq t < t_{o(i+1)}$ (time validity requirement). The polynomial choice is to meet the computation requirement, it is efficient to build in live using EJML [Abe10]. Moreover, from the error requirement, we derive the following continuous segmentation condition: $\forall j, |f_{c_{ij}}(t) - y_{c_{ij}}(t)| < \varepsilon_{c_{ij}}$. where $y_{c_ij}(t)$ is the physical measured value of the attribute c_{ij} at time t, and $\varepsilon_{c_{ij}}$ the maximum tolerated error of this attribute as defined in the meta model.

From the continuity requirement, we get the initial condition of the polynomial functions when starting a new segment: $\forall i, \forall j, f_{ij}(t_{oi}) = f_{(i-1)j}(t_{oi})$. This leads to the following fact: $\forall i, \forall j, w_{ij0} = f_{(i-1)j}(t_{oi})$. Meaning that the constant part of the polynomial is always equal to the latest value generated by the polynomial of the previous segment, for all segments i and for all continuous attributes j.

The self-containment requirement can be easily checked by the fact that the polynomial uses only the weights w_{ijk} and the time origin t_{oi} in its calculation. All this information is stored within the segment.

Finally, each segment can be serialized and de-serialized. It forms the payload to be stored/retrieved in/from a database as presented in figure 6.3. Note that, our design to encode all attributes in the same segment is to reduce the total number of segments created. This way, if many attributes change at the same time, all of the changes will be encoded

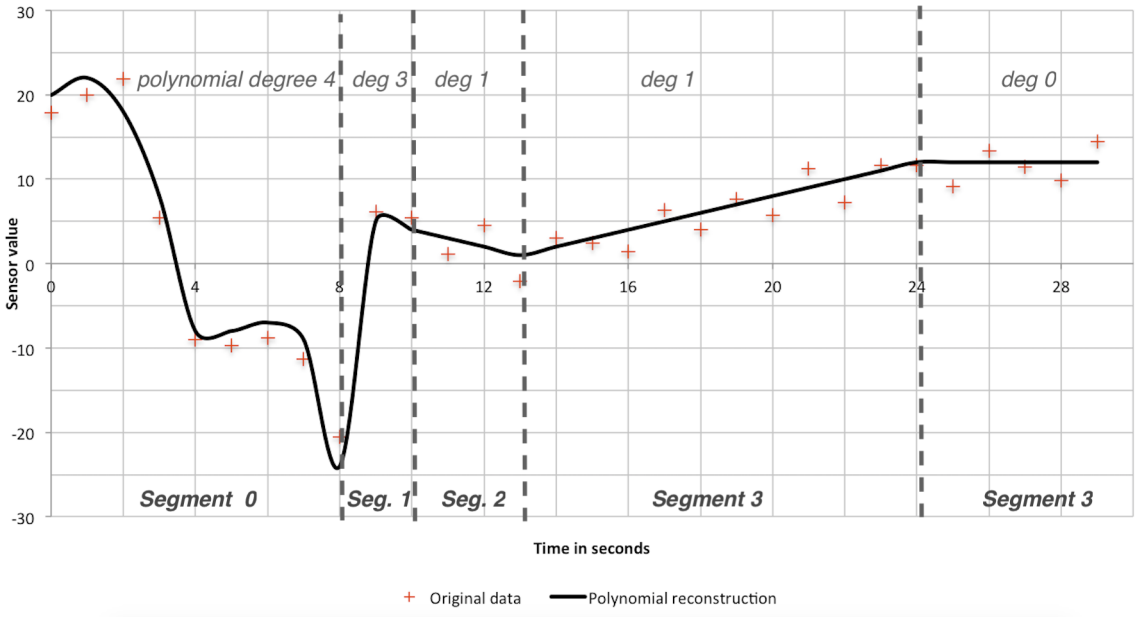

Figure 6.4: A segmentation and polynomial encoding of a continuous attribute

in one segment rather than in many segments, to speed up the lookup on the index tree. Whenever a get or set operation is done on a model element, the index tree structure finds the closest segment to the requested time. This has a complexity of $O(log|S|)$ where $|S|$ is the number of segments present in our model. Then the segment delegates the get/set operation to our algorithm that we present in the next section.

6.3.3 Online Segment Construction Based on Live Machine Learning

In this section, we present the algorithm that takes care of deciding the degree of the polynomials, calculating their weights, segmenting and storing them whenever the previously discussed requirements are not met. All this should be done in an on-line live learning manner, completely transparent for the final users and while guaranteeing the maximum tolerated error ε for each continuous attribute. Figure 6.4 shows an example of a segmentation of a continuous attribute with the real points that were used to build the segments and the different polynomial representations.

The main functionality of our algorithm is in the insert function presented in listing 1. In a nutshell, when the first sensor sample is fed to our model, a polynomial of degree 0 is created from this sample. As long as new values fit in the current polynomial function within the maximum tolerated error for the current degree, the current polynomial model is maintained. When a value does not fit in the current model, we check first if we can increase the polynomial degree, i.e., if (degree+1) is lower than the maximum degree *MAXDEG*. If this is the case, we regenerate a temporary set of samples from the current polynomial, and add the newly arriving value to this set, and we try to reconstruct a new polynomial of a higher degree (degree +1) using EJML linear solver [Abe10]. We send to this linear

solver the live generated dataset, and the desired polynomial degree, it returns back the coefficients of the polynomial that fits the best the dataset.

We then do a check to test if the new polynomial represents the generated dataset within a certain tolerated error which depends on the maximum tolerated error specified by the user and the current degree of the polynomial. If this is the case, we update the polynomial to the new representation. If not, this means that increasing the degree will not guarantee that the representation fit within the maximum tolerated error specified by the user, or the polynomial has reached the $MAXDEG$. So the new sample does not fit in the previous polynomial and we were unable to increase its degree, thus the only solution is to create a new segment. In this case, the previous polynomial function is stored, and a new polynomial function is created from the new sample which could not fit previously, and the process restarts.

Algorithm 1 *Polynomial_fit*

```
void insert(long time, double value) {
/*if null init a polynomial of degree 0*/
if (weights == null) {
deg=0;weights = new double[1];
initTime=time; weights[0]= value;
}
/*test if the sample fits in previous function*/
if (Math.abs(get(time) - value) <= maxError(deg)){
return;
}
/*if not, increase previous function degree*/
do {
deg++;
double[] times = new double[deg+1];
double[] values = new double[deg+1];
/*Regenerate values from previous polynomial*/
for(int i=0; i<deg; i++){
times[i]= initTime + (i/deg)*(time-initTime);
values[i]= get(times[i]);
}
/*add the new sample*/
times[i] = time; values[i] = value;
/*fit a new polynomial of a higher degree*/
pf = new PolynomialFitEjml(deg);
pf.fit(times, values);
//test if the polynomial fits within maxError
if(maxError(pf.getCoef())<=maxError(deg)){
/*if yes, we keep this new function*/
weights = pf.getCoef();
return;
}
} while(deg<MAXDEG);
/*if degree doesn't fit within MAXDEG,*/
/*a new segment will be created*/
segments.save(initTime, weights);
/*reset the process*/
weights=null; insert(time,value);
}
```

Our model guarantees that the polynomial representation does not deviate from the real value more than a maximum tolerated error ε provided by the user. However, because we are constructing the polynomial on the fly, starting from a degree 0 and upgrading the degree step by step, the error cumulates. Thus we need to define a maximum tolerated

error strategy $e(d)$ for each step of the construction (aka. when the polynomial is of a degree d), in a way that the total cumulated error is ensured not to exceed the maximum tolerated error defined by the user. In other words: $\sum(e(d)) \leq \varepsilon$.

Another requirement is that the more complex the polynomial representation becomes, the less deviation from the previous representation, we will be ready to tolerate. Otherwise, the total cumulated error will not converge. In mathematical terms: $e(d+1) < e(d)$.

Combining these 2 requirements, our design choice is the following: $e(d) = \varepsilon/(2^{d+1})$. One can easily verify both properties: First it is a constantly decreasing function, and second the sum of the error will be always less or equal than ε because $\sum(1/(2^{d+1})) = 1$.

In this way, when the polynomial representation is a constant (degree 0), the model can deviate maximum of $\pm\varepsilon/2$ from the real values. When we increase the degree to 1 (a line), we can tolerate $\pm\varepsilon/4$ more error deviation. The cumulated error for this step is less than $\pm 3\varepsilon/4$. And so forth, if the degree of the polynomial reaches infinity, the maximum deviation between our model and the reality will be $\pm\varepsilon$, the one specified by the user in the meta model.

6.4 Experimental Evaluation

The main goal of our contribution is to offer an efficient models@run.time structure to deal with IoT related data. Thus, this evaluation section puts the emphasis on the performance of our modeling structure and its ability to imitate and rebuild the initial continuity of IoT data. As key performance indicator *(KPI)* we selected classical **write**, **sequential read**, **random read** operations and **storage size**. In addition, we introduce the KPI **continuity reconstruction ability *(CRA)***, which aims to evaluate the ability of the modeling layer to rebuild a continuous signal – even in case of sampling rate variation of one model element. To conduct these experiments, we reused our past work [HFN+14a], which has been demonstrated to be highly scalable following a discrete strategy. We enhanced this implementation with the continuous polynomial based strategy, and evaluated it on various IoT data sets, with and without this new optimization. The rest of this section is organized following the above defined KPIs, and in addition we add a subsection to show that these results are valid, regardless of the chosen data store implementation *(Google LevelDB or MongoDB)*, up to a common multiplicative factor.

Experimental Setup:

We integrated our optimization approach into the Kevoree Modeling Framework [i] *(KMF)* and all experiments have been conducted on the latest java version 8. The implementation of our approach is available as open source [ii]. We implemented two live learning segmentation strategies. First, with Piecewise Linear Representation *(PLR)*, which is, to the best of our knowledge, the most efficient implementation for time-series compression, and second with our polynomial extended version. The experiments for all KPIs have been carried

[i] http://kevoree.org/kmf/
[ii] https://github.com/dukeboard/kevoree-modeling-framework

out with KMF using Google's LevelDB [iii] as a database. To connect KMF to LevelDB we implemented a corresponding driver. The only exception of this is the database impact benchmark at the end of this section, where we use in comparision MongoDB [iv] as a database. Again, in order to connect KMF to MongoDB we implemented a corresponding driver. In addition, for all experiments we used the models@run.time in a stream mode, meaning that values are propagated as soon as available. A batch strategy could offer a better performance for raw inserting of values but would introduce a delay, which is in many IoT systems unacceptable. Nonetheless, in the last section of this evaluation we present results for both batch and stream modes, to demonstrate that our approach offers the same advantages for batch and stream (excluding a multiplication factor).

As argued before, IoT data have very specific characteristics that make them challenging to store and process. In order to properly represent these characteristics we selected seven different datasets, containing data from the best to the worst case for our approach. All measurements have been carried out on an Arduino platform and all of them have been normalized to **5 000 000** entries. Table 6.1 presents a summary of the datasets and their properties:

Database	Sensor
DS1: Constant	c=42
DS2: Linear function	y=5x
DS3: Temperature	DHT11 (0 50'C +/- 2'C)
DS4: Luminosity	SEN-09088 (10 lux precision)
DS5: Electricity load	from Creos SmartMeters data
DS6: Music file	2 minutes samples from wav file
DS7: Pure random	in [0;100] from random.org

Table 6.1: Experimental Selected Datasets

The first two datasets are artificial and represent the best two case scenarios: a sensor measuring the same value overtime, or a linearly growing measure. The third and fourth datasets are measured from temperature and luminosity sensors, like the ones that can be found in personal weather stations. The fifth dataset is issued from our industrial partner Creos Luxembourg S.A. which conducted experiments in three regions in Luxembourg to measure the electric consumption based on smart meters. The sixth dataset is an open source music file, which has been sampled into chunks of amplitudes. Finally, the seventh dataset is a purely randomly generated set of doubles.

The selected datasets highlight three very different types of IoT data. DS1 and DS2 are aperiodic (nearly constant) signals, while DS3, DS4, and DS5 contain periodic signals *(hourly, daily...)* and finally DS6 and DS7 are (almost) random signals.

All experiments have been carried out on a Core i7 computer equipped with 16GB RAM and SSD drive.

[iii] http://leveldb.googlecode.com/svn/trunk/doc/benchmark.html
[iv] http://www.mongodb.org

KPI Write Evaluation:

In this first benchmark we evaluate execution time to insert data in a stream mode. Therefore, we use the data from our datasets DS1 to DS7. We run the experiments three times per dataset. The results are depicted in figure 6.5.

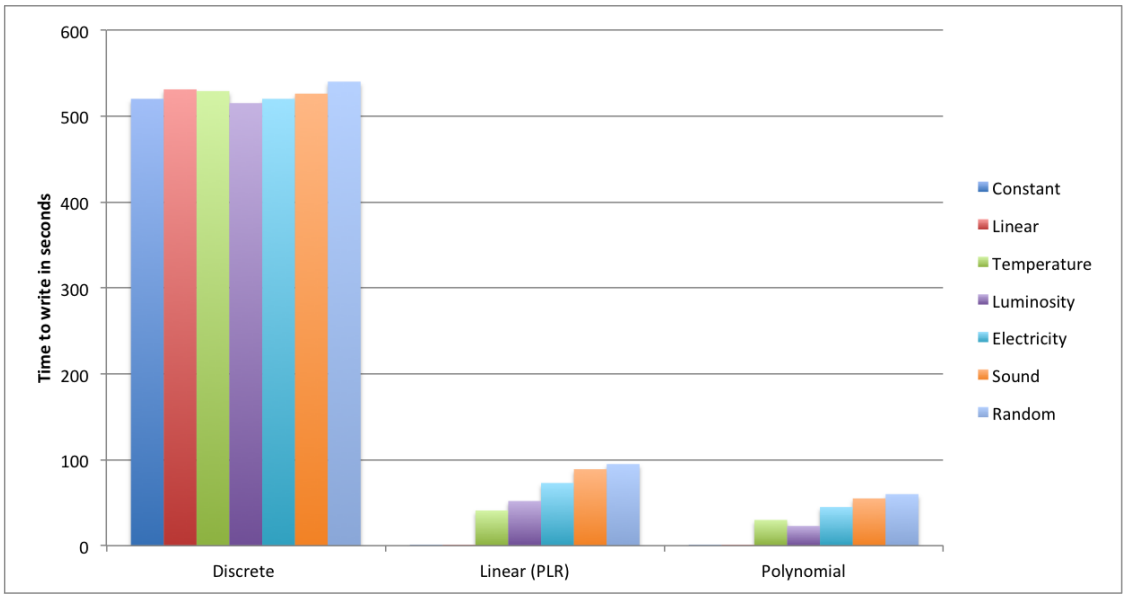

Figure 6.5: Time to write on Leveldb for the 3 methods

The evaluation criteria for this KPI is execution time, which is represented on the vertical axis. On the horizontal axis we depict all datasets organized by the segmentation method. As segmentation methods we use discrete, PLR linear, and our polynomial-based live compression method. All values are saved in a database (SSD drive).

The discrete modeling method needs roughly the same time to write for all datasets. This is not a surprise since this time depends only on the number of samples, which has been normalized to 5 000 000 elements for all datasets. The observed small variations can be explained by the different numbers of bytes to represent each value *(mainly depending on the precision of the signal)*. For the linear and polynomial representations we observe very different times for the different datasets. This is because the time depends on the number of generated segments, which itself depends on the dataset. The constant dataset results in the fastest time and the random dataset in the worst time.

However, we observe that in all cases the polynomial is faster to write compared to a complete discrete method. In fact, the polynomial is very close (or even similar) to a PLR method. Through this benchmark we demonstrate that the time spent to compute polynomial representations and compress the signal is still below the time to store all enumerated values, even on a fast SSD drive.

KPI Sequential Read Evaluation:

In this benchmark we evaluate the time to read a sequence of elements from the model. We select this operation for the KPI sequential read because it is the most used operation

in machine learning or pattern-detection algorithms that need to access model elements sequentially to compute a user profile or a similar trend. For instance, in domains such as smart grids, the sequential read is mostly used to extrapolate electric consumption data that feed an alert system. The sequential read is thus performance critical for IoT oriented models@run.time. Again we run the experiments for DS1 to DS7 and use the three live compression methods. Results are depicted in figure 6.6.

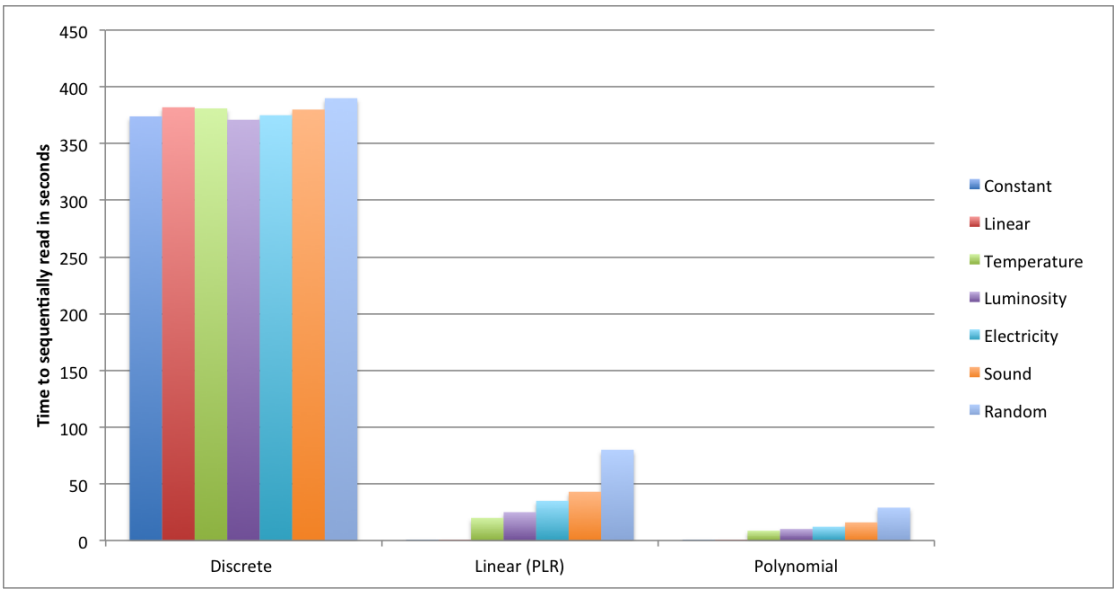

Figure 6.6: Time to read sequentially from Leveldb for the 3 methods

Again, the vertical axis represents the time to read all dataset values and the horizontal axis depicts the different datasets and method combinations. Similarly to the write benchmark, the discrete method introduces very few variations in time due to the normalized datasets. However, we can observe a significant improvement factor, between discrete and, PLR or polynomial representations. We can also notice a significant improvement of polynomial versus PLR. The polynomial can read thousand times faster for the best cases: constant or linear dataset. This can be explained by the fact that both, the PLR and polynomial models encode the constant or the linear function in one function. For the other datasets the average speedup in sequential read is between 30-50 times faster. This significant improvement factor can be explained by the fact that a compressed representation can contain many points, reducing both the amount of database calls but also the size of segment indexes. Moreover, the polynomial capacity is much better than the linear one. This explains why we can observe a speedup between 2-3 times between both approaches.

KPI Random Read Evaluation:

In this section we conduct benchmarks to evaluate the Random Read KPI. Here we are evaluating the capacity of models@run.time to offer scalable read operations for elements randomly distributed in the model itself. This operation is a good example for applications that need to lookup random elements (ex. Monte-Carlo simulations). Again this operation is widely used in intelligent system and performance is critical. We use the same experimental

setup than for the sequential read, except the model element selection simulates a random process. Results are presented in figure 6.7.

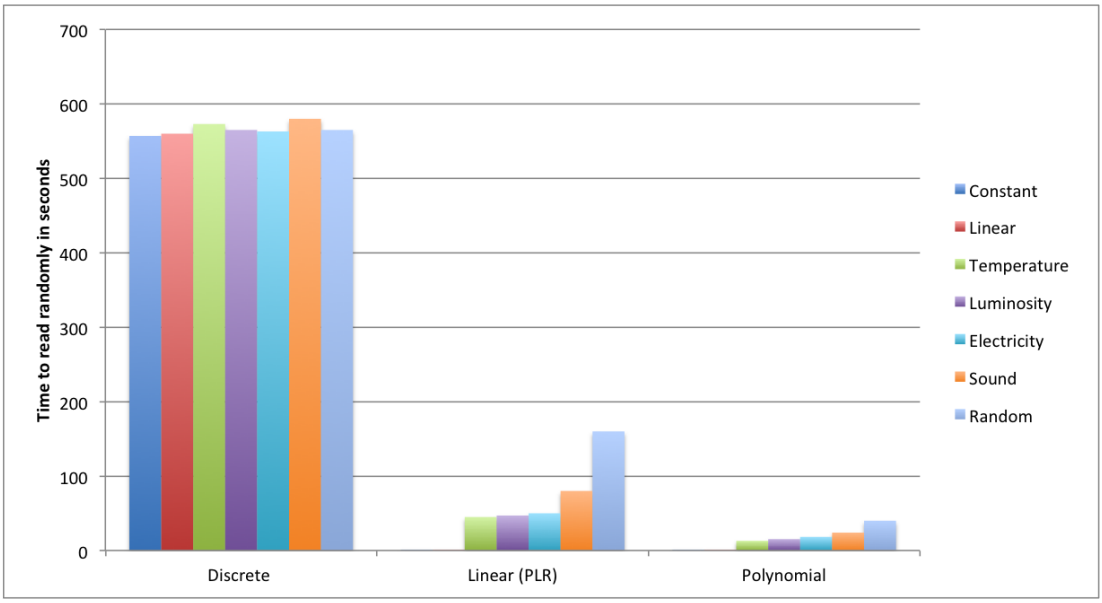

Figure 6.7: Time to read randomly from Leveldb for the 3 methods

Again, results are similar to the sequential read, except that the PLR performs worse than for the sequential read. In contrast, the polynomial encoding can still be up to thousand times faster for the constant or linear dataset. For the other datasets, the average speedup is between 40-60 times.

KPI Storage Size Evaluation:

In this section we conduct experiments to evaluate the gain in terms of storage space. Indeed, both the PLR and polynomial implementations try to reduce the number of segments and thus reduce the overall models@run.time storage requirements. Using the same setup used for *KPI Write*, we observe two things for the *KPI Storage Size*. First of all, the number of segments after live insertion into the models@run.time. This number reflects the pure compression of the signal in memory. Secondly, we measure the amount of bytes, which are sent to the NoSQL storage through the models@run.time layer. This evaluation takes the overhead of models into account and validates the performance in an integrated environment. In table 6.2 we present the segmentation results.

In this result we are considering the 5 000 000 elements inserted into the discrete method. The compression rate is computed according to the size to represent a long value in the Java Virtual Machine. In other words, this compression rate is equivalent to the memory saved by the virtual machine to represent the signal, expressed in bytes. For the polynomial method, on a constant or linear signal the compression is maximal, at around 99%. This compression is still between a range of 73 to 46% for IoT datasets and fall at 33% and 10% for random signals, which are our worst case scenarios.

Database	# Seg. PLR	Rate	# Seg. Poly	Rate
Constant	1	99.99 %	1	99.99 %
Linear fct	1	99.99 %	1	99.99 %
Temperature	316,981	93.66 %	78,580	73.54 %
Luminosity	2,554,227	48.91 %	296,362	63.44 %
Electricity	3,966,521	20.66 %	396,209	46.42 %
Music file	4,508,449	9.83 %	461,603	33.21 %
Random	4,999,824	0.01 %	682,324	10.53 %

Table 6.2: Number of segments after live insertions and rate

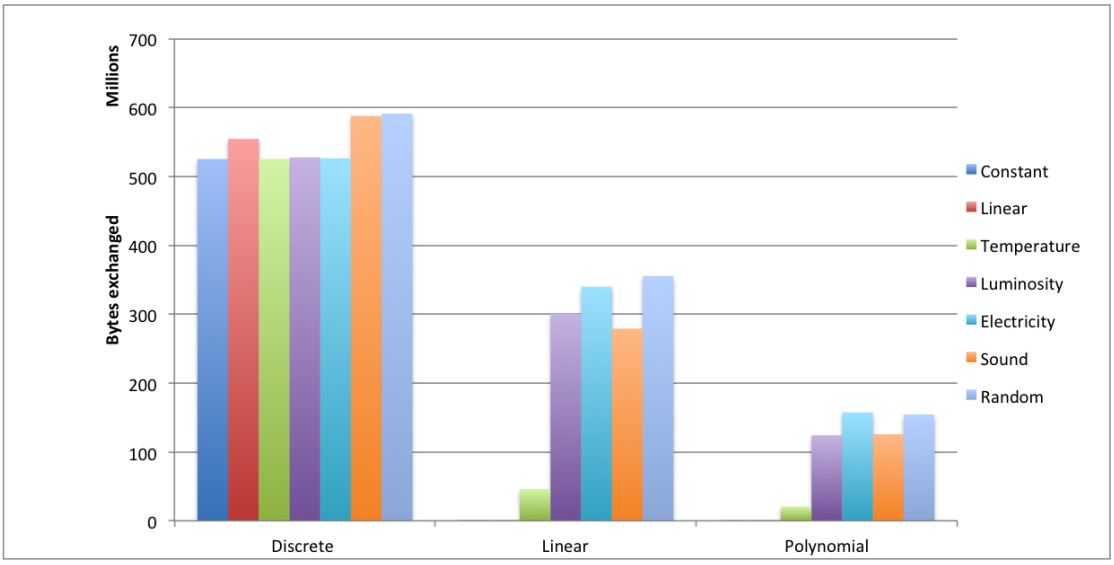

Figure 6.8: Bytes exchanged between NoSQL storage and models during 5 000 000 write operations

In this graph (figure 6.8) the vertical, respectively horizontal axis represent the bytes stored in the database and the combination of methods and datasets. We observe a similar trend like when we operate in-memory, however, the compression rate, in average, is less efficient. This is due to the overhead of the serialization strategy, which converts long into a byte representation in order to be transferred to the database layer. In fact, polynomials use high precision double numerical values, leading to very long string representations, which makes the compression rate slightly weaker. This overhead introduced by the implementation of KMF can be optimized in work by using a binary serialization format.

KPI Continuity Reconstruction Ability Evaluation:

In this experiment, we measure the ability of the model to reconstruct the value-continuity. The goal is to be as close as possible to the original signal in case of samples loss. The motivation behind this experiment is based on sensor network properties. In fact, due to network errors or simply in order to reduce energy consumption, sensors can change their sampling rate. Here, we evaluate the capacity of our continuous model to detect such missing values and extrapolate them using the various segmentation strategies. We engage the same experimental setup than the KPI Write benchmark, except that we uniformly

drop an average of one value out of 10. Later, in an experiment similar to the sequentially read, we reconstruct the missing samples and we measure the error *(deviation from the real missing value)*. Our results are presented in table 6.3.

Database	Discrete	Linear	Polynomial
DS1: Constant	0%	0%	0%
DS2: Linear function	5 %	0%	0%
DS3: Temperature	8.5%	3%	3%
DS4: Luminosity	9.9%	3.6%	3.5%
DS5: Electricity	17 %	7%	6%
DS6: Sound sensor	21%	15%	13%
DS7: Random	31.8%	31.1%	30.8%

Table 6.3: Average error for both structures when we try to approximate missing values

Each extrapolation introduces a deviation, however through our experiments we highlight that the polynomial strategy compensates best for lost samples. For the random dataset, all methods lead to almost the same recovery error due to the random nature of the dataset. The PLR approach was already better than the discrete strategy, which imitates more or less a square signal. However, in all cases the polynomial method offers a better trade-off and is more resilient to data loss.

Results Generalization:

As discussed before in the experimental setup subsection, the database used for the experiments and the mode *batch or stream* influence the absolute values for the various conducted benchmarks in a significant manner. In this last benchmark we evaluate experimentally this threat to validity by running the same write operations on various configurations. We run the *KPI Write* benchmark two additional times in *stream* and *batch* mode, once on top of the *LevelDB* driver and the *MongoDB* driver. Results are presented in table 6.4.

Test	LevelDB	MongoDB
Stream Discrete	500 to 520s	680 to 750s
Stream Polynomial	1 to 60s	1 to 80s
Batch Discrete	70 to 75s	77 to 84s
Batch Polynomial	1 to 10s	1 to 15s

Table 6.4: Time range in seconds for save in LevelDB and MongoDB

Results are presented in time (seconds) and in range (from best to worst case based on the different datasets). As expected, batch processing offers better performance to insert 5 000 000 model elements and this regardless of the chosen database implementation. Comparing the two databases it is interesting to notice that the performance difference of MongoDB and LevelDB is consistent, at around 33% to 45%. We argue that the chosen database can only impact our results with a multiplication factor. In the same manner the ratio between polynomial and discrete is also consistent between the stream and batch mode, with a factor of roughly 11 to 17. We argue here that the mode only impacts with a multiplication the

final results. Overall, we can assess the validity our results if considering the improvement factor and not raw values, regardless of the chosen mode or database.

Experimental Results Summary:

In this section we have conducted various benchmarks to demonstrate the general application of our polynomial segmentation method for models@run.time optimization of IoT data. The theoretical limit of our polynomial approach is based on the maximal polynomial degree constructed during the segmentation method. During all this experiment we demonstrated that our chosen strategy with a **maximal degree of 20** applies the best for all the evaluated datasets. We also argue that such strategy improves both time and value continuity.

6.5 Discussion: Data-Driven Models or Model-Driven Data?

While models@run.time and modeling in the large techniques are progressing further and further, the border between databases and models tends to blur more and more. This trend is even more reinforced by recent NoSQL techniques and other storage systems for unstructured data, which leave to other software layers the freedom to define the semantic and the structure of data. This enables data storages to efficiently scale horizontally, as can be seen in many Big Data approaches.

The vanishing border between models and data storage systems inspires this discussion about the role of data storage systems and models. Our contribution reflects this trend since models@run.time becomes a model-driven data structure, supporting the creation of data-driven intelligent systems. This pushes the management of data partly into the application layer and allows to use simpler storage solutions, like those developed for Big Data storage systems.

Beyond structural information, a model can define a rich semantic, like our definition of continuity for attributes. Leveraging this additional information, a model can organize and process data in a very efficient way, like we suggest with our polynomial approach. With traditional data-schemas a storage can hardly perform such optimizations, because it only stores a stream of values. Although it optimizes the storage, our objective is to offer a rich semantic to data structures, which is one of the primary goals of models.

6.6 Conclusion

The Internet of Things and smart systems with their reasoning abilities are considered to be one of the next disruptive usages of computers in the future. To tackle the complexity of reasoning tasks, these systems leverage techniques like models@run.time to structure and process measured information in efficient context representations. However, today's computation theory is based on the discretization of observable data into a sequence of events, augmented by timestamps. Nevertheless, the amount of expected data in the IoT domain makes the construction of reasoning contexts through the enumeration of all timestamped

measurements challenging. Beyond models@run.time, this discrete representation for IoT is depicted by Jacobs *et al* [Jac09] as the *pathology of Big Data*.

We claim that a discrete data representation is not optimal for all attribute types in models, especially for data which is naturally continuous. Since models should be *simpler, safer and cheaper than the reality* [RWLN89] we introduce a new meta attribute type for continuous IoT related data. By leveraging our previous results on model element versioning and the state-of-the-art time series compression algorithms, we enable models@run.time to handle millions of volatile data aside traditional discrete data such as string and relationships. We enhanced and adapted signal processing and time series techniques to allow models to go beyond value enumeration capabilities. We integrated our algorithm based on polynomial representation into the Kevoree Modeling Framework to seamlessly and efficiently store continuous values in any models.

During an experimental validation we demonstrated the suitability of this approach to build context models handling millions of values from various IoT datasets. The various benchmarks conducted in this work, lead to the conclusion that this representation can enhance, with a very significant factor (from 20 to 1000), primary operations for intelligent systems such as read or write over a context model.

In this chapter, we enabled one AmI agent to model and represent efficiently his knowledge base from the information coming from the sensors. The next step, is to enable each agent to communicate with other agents in order to derive the contextual information in a distributed way.

7

R-Core: a rule-based contextual reasoning platform for AmI

After enabling local agents to represent continuous sensor data efficiently in a local knowledge base using data models, the next contribution is to enable distributed context awareness using a component-based model architecture that allows AmI agents to intercommunicate. In this chapter, we present a framework that applies ideas from contextual defeasible logic for AmI environments. We show how to derive context information in a complete distributed way in a use case study from the AAL domain.

This chapter is based on the work that has been published in the two following papers:

- Assaad Moawad, Antonis Bikakis, Patrice Caire, Grégory Nain, and Yves Le Traon. R-core: A rule-based contextual reasoning platform for ami. In *RuleML@ ChallengeEnriched 2013*, 2013
- Assaad Moawad, Antonis Bikakis, Patrice Caire, Grégory Nain, and Yves Le Traon. A rule-based contextual reasoning platform for ambient intelligence environments. In *Theory, Practice, and Applications of Rules on the Web*, pages 158–172. Springer, 2013

Contents

7.1 Introduction

Augmenting the environment with sensing capability and efficiently modeling the physical attributes is the first step to enable reasoning in AmI. In this chapter, we discuss the next step which is enabling context awareness over a distributed component-based model architecture. This contribution is to enable reasoning with contextual defeasible logic (CDL) and to apply it in a real AmI environment. In this chapter, we focus on scenarios from *Ambient Assisted Living* (AAL), though the same findings may be applied to other subfields of AmI. AAL is a relatively new research and application domain focused on the technologies and services to enhance the quality of life of people with reduced autonomy, such as the elderly. In the framework of the CoPAInS project[i], such problematic is studied in order to derive contextual information [MEC+12a].

Creating this bridge between CDL and AAL requires addressing issues, such as dynamicity, adaptability, detection and communication between devices, and is therefore not trivial. This prompts our research question: *How to deploy CDL in real AmI environments?* To address this question, we created a platform that maps the CDL model to the business view of AAL using component-based models deployed on Kevoree [FBP+12]. The contribution of this work is twofold: (a) we describe solutions for the deployment of a theoretical model (CDL) in real AmI environments; and (b) we provide an AAL platform, usable by anyone to test and implement AAL scenarios.

In this chapter, we address the distributed context awareness problem of the big picture, as presented in figure 7.1. From a combination of local knowledge and remote knowledge coming from other AmI agents, how can we enable context awareness using contextual defeasible logic?

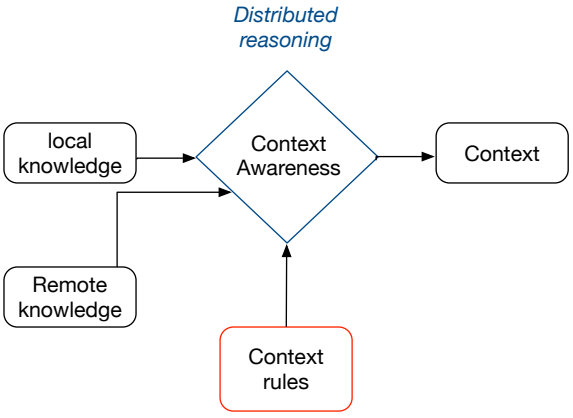

Figure 7.1: Distributed context awareness.

[i] http://wwwen.uni.lu/snt/research/serval/projects/copains

7.2 An Ambient Assisted Living Example

In this section, we present an Ambient Assisted Living (AAL) scenario, part of a series of scenarios validated by HotCity, the largest WI-FI network in Luxembourg, in the Framework of our CoPAInS project (Conviviality and Privacy in Ambient Intelligence Systems).

In our scenario, visualized in figure 7.2, the eighty-five years old Annette is prone to heart failures. The hospital installed a Home Care System (HCS) at her place. One day, she falls in her kitchen and cannot get up. The health bracelet she wears gets damaged and sends erroneous data, e.g., heart beat and skin temperature, to the HCS. Simultaneously, the system analyzes Annette's activity captured by the Activity Recognition Module (ARM). Combining all the information to Annette's medical profile, and despite the normal values transmitted by Annette's health bracelet, the system infers an emergency situation. It contacts the nearby neighbors asking them to come and help.

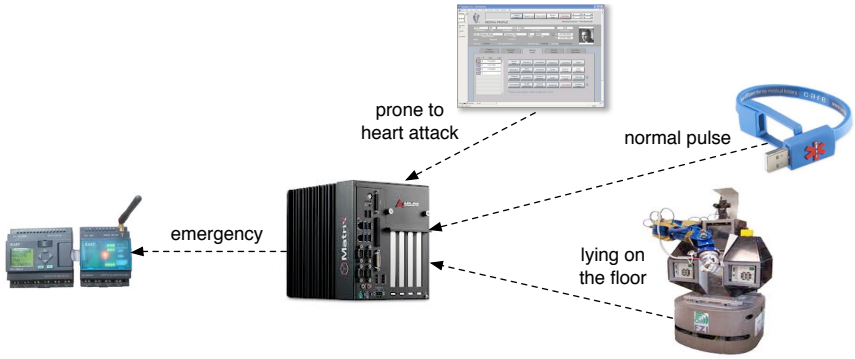

Figure 7.2: Context Information flow in the scenario.

This scenario exemplifies challenges raised when reasoning with the available context information in ambient intelligence environments. Furthermore, it highlights the difficulties in making correct context-dependent decisions.

First, context knowledge may be erroneous. In our example, the values transmitted by the health bracelet for Annette's heart beat and skin temperature, are not valid, thereby leading to a conflict about Annette's current condition. Second, local knowledge is incomplete, in the sense that none of the agents involved has immediate access to all the available context information. Third, context knowledge may be ambiguous; in our scenario, the HCS receives mutually inconsistent information from the ARM and the health bracelet. Fourth, context knowledge may be inaccurate; for example, Annette's medical profile may contain corrupted information. Finally, devices communicate over a wireless network. Such communications are unreliable due to the nature of wireless networks, and are also restricted by the range of the network. For example, the health bracelet may not be able to transmit its readings to HCS due to a damaged transmitter.

Example. The scenario described in section 7.2 may be modeled as follows in CDL. We consider 5 different contexts: *sms* for the SMS system, *hcs* for the Home Care Syste, *arm* for the activity recognition module, *br* for the bracelet, and *med* for the medical profile.

sms has only one mapping rule according to which, when the Home Care System detects an emergency situation, the SMS system dispatches messages to a prescribed list of mobile phone numbers:

$r^m_{sms} : (hcs : emergency) \Rightarrow (sms : dispatchSMS)$

The Home Care system imports information from the activity recognition module, the bracelet and Annete's medical profile to detect emergency situations using two mapping rules:

$r^{m1}_{hcs} : (br : normalPulse) \Rightarrow \neg(hcs : emergency)$

$r^{m2}_{hcs} : (arm : lyingOnFloor), (med : proneToHA) \Rightarrow (hcs : emergency)$

The factual knowledge of the other three modules is modeled using local rules with empty body:

$r^l_{br} :\rightarrow (br : normalPulse)$

$r^l_{arm} :\rightarrow (arm : lyingOnFloor)$

$r^l_{med} :\rightarrow (med : proneToHA)$

The Home Care System is configured to give highest priority to information imported by the medical profile and lowest priority to the bracelet:

$T_{hcs} = [med, arm, br]$

7.2.1 Distributed Query evaluation

In [BA10], Bikakis et al. presented an argumentation semantics of CDL, while in [BAH11] they provided four algorithms for query evaluation. $P2P\_DR$ is one of these algorithms, which is called when a context C_i is queried about the truth value of one of its local literals $(c_i : a_i)$, and roughly proceeds as follows:
We should note that a rule r_i is applicable when for all its body literals we have obtained positive truth values. SS_{r_i} is the union of the foreign literals with the Supportive Sets of the local literals contained in the body of r_i, while $SS_{(c_i:a_i)}$ is the *strongest* between the Supportive Sets of the rules with head $(c_i : a_i)$. A set of literals S_1 is *stronger* than set S_2 w.r.t. T_i *iff* there is a literal l in S_2, such that all literals in S_1 are stronger than l w.r.t. T_i. A literal $(c_k : a)$ is stronger than literal $(c_l : b)$ w.r.t. T_i *iff* C_k precedes C_l in T_i.

Example (continued). In our running example, a query to *sms* about $(sms : dispatchSMS)$ will initiate a second query to *hcs* about $(hcs : emergency)$. The second query will in turn initiate three more queries: a query to *br* about $(br : normalPulse)$; a query to *arm* about $(arm : lyingOnFloor)$; and a query to *med* about $(med : proneToHA)$. $P2P\_DR$ will return *true* for the latter three queries, and will compute $SS_{(hcs:emergency)} = \{(arm : lyingOnFloor), (med : proneToHA)\}$ and $SS_{\neg(hcs:emergency)} = \{(br : normalPulse)\}$.

Algorithm 2 *P2P_DR*

if $(c_i : a_i)$ (**or** $\neg(c_i : a_i)$) is derived as a conclusion of the local rules of C_i **then**
  return *true* (or *false* resp.)
else
  for all rules r_i in c_i that have $(c_i : a_i)$ or $\neg(c_i : a_i)$ in their heads **do**
    if r_i is applicable **then**
      Compute the Supportive Set of r_i, SS_{r_i}
    end if
  end for
  Compute the Supportive Sets of $(c_i : a_i)$, $SS_{(c_i:a_i)}$, and $\neg(c_i : a_i)$, $SS_{\neg(c_i,a_i)}$
  if $SS_{(c_i:a_i)}$ is *stronger* than $SS_{\neg(c_i,a_i)}$ with respect to T_i **then**
    return *true*
  else
    return *false*
  end if
end if

W.r.t. T_{hcs}, all elements of $SS_{(hcs:emergency)}$ are stronger than $(br : normalPulse)$, therefore $P2P\_DR$ will return a positive truth value for $(hcs : emergency)$, and the same value for $(sms : dispatchSMS)$ too, as there is no rule that supports its negation.

As shown in the example above, CDL may deal with several of the challenges of ambient intelligence environments, such as uncertainty, ambiguity, and erroneous data. There are still, though, some questions that need to be addressed in order to fully deploy CDL in real environments: how do the devices actually detect and communicate with each other? and how can we achieve dynamicity and adaptability? In the next sections, we describe how we addressed such questions by integrating CDL in the software platform of Kevoree.

7.3 R-CoRe Architecture

In this section, we describe how CDL and Kevoree are integrated in the R-CoRe architecture. We should note that the parts of CDL, which were not directly mapped to existing Kevoree elements, were implemented in Java.

7.3.1 Java Implementation

Our Java implementation is composed of a main rcore package containing 4 sub-packages: agencies, interceptor, knowledge, logic packages and the main Query component class.

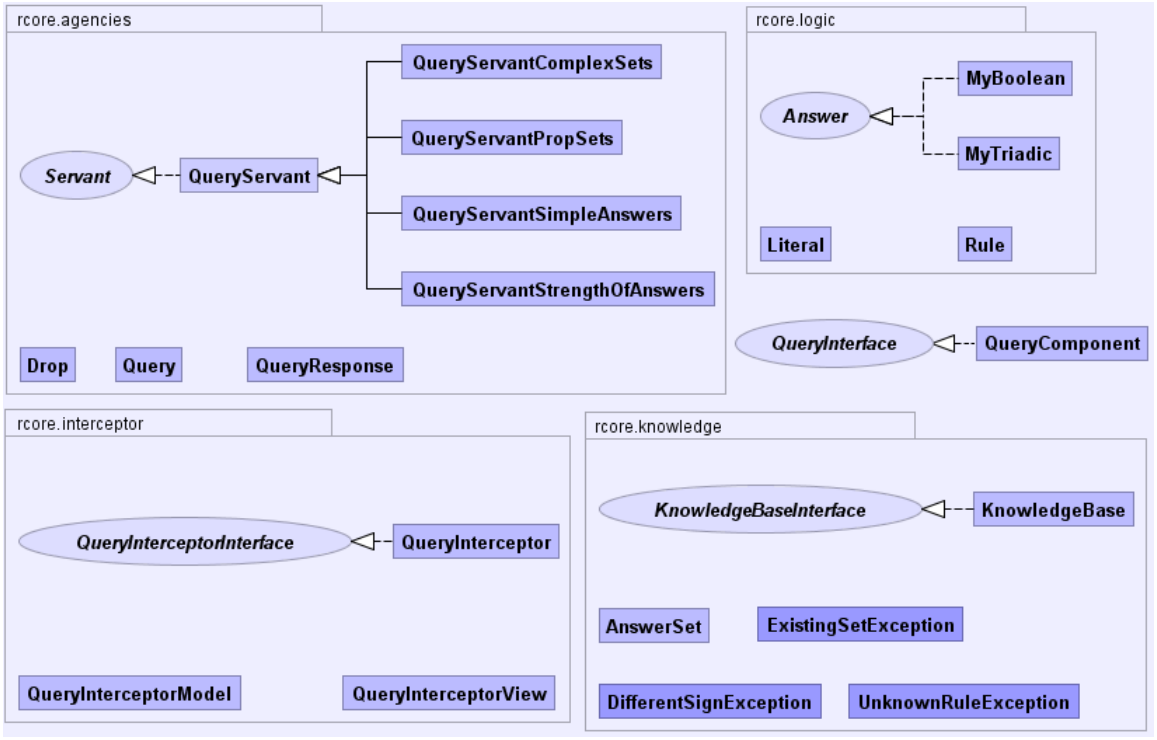

Figure 7.3: General overview of R-CoRe.

- The agencies package contains the classes used by the query servant thread, which implements the reasoning algorithms of CDL [BAH11]. This package contains also the Query class and the QueryResponse class. These classes represent the massages that will be exchanged between QueryComponents.
- The interceptor package contains the model, the view and the controller of the interceptor component. The controller is a component developed to run on the Kevoree platform.
- The knowledge package includes the KnowledgeBase class, which stores the local rule theory, the mapping rules and the preference order of each node. This class contains also the methods used to load and save the knowledge base and preference order from and to files.
- The logic package contains the classes that represent (in memory) the literals and rules.
- Finally the query package contains the QueryComponent class, which is the main component developed to run on the Kevoree platform.

Figure 7.3 shows a UML overview of our implementation.

7.3.2 Query Component

In our platform, the notion of context, is implemented by a new component type that we developed, called *Query Component*. This component has two inputs: *Console In* and

Query In, and two outputs: *Console out* and *Query Out*. The Query Component has three properties: a *Name*, an *initial preference address* and an *initial knowledge base address*. In Kevoree, each instance must have a unique name. In R-CoRe, we use this unique name to specify the sender or the recipient of a query. The preference address and the knowledge base address contain the addresses of the files to be loaded when the component starts. The knowledge base file contains the rule set of a context, while the preference file contains the preference order of the context implemented as a list.

Each component has two console (in/out) and two query (in/out) ports. The console input port is used to send commands to the component, e.g. to update its knowledge base or change its preference order. The outputs of the commands are sent out to the console output port. The query in/out ports are used when a component is sending/receiving queries to/from other components. Queries are sent via the “Query out" port and responses are received via “Query In".

Internally, the Query Component has some private variables, which represent its knowledge base, the preference order and a list of query servant threads currently running on it. When the component receives a new query, it creates a new query servant thread dedicated to solve the query and adds it to the list of currently running query threads. When this thread reports back the result of the query, it is killed and removed from the list.

7.3.3 Query Servant

When a query servant thread is created, it is always associated with an ID and with the query containing the literal to be solved, and it is added to the list of running threads of the query component. The query servant model works as follows:

1. The first phase consists of trying to solve the query locally using the local knowledge base of the query component. If a positive/negative truth value is derived locally, the answer is returned and the query servant terminates.

2. The second phase consists of enumerating the rules in the knowledge base that support the queried literal as their conclusion. For each such rule, the query servant initiates a new query for each of the literals that are in the body of the rule. For foreign literals, the queries are dispatched to the appropriate remote components. After initiating the queries, the query servant goes into an idle state through the java command “wait()”.

3. When responses are received, the query servant thread is notified. Phase two is repeated again, but this time using the rules that support the negation of the queried literal.

4. The last step is to resolve the conflict by comparing the remote components that were queried for the two literals using the context preference order. The result is reported back to the query component.

7.3.4 Query Interceptor

In order to monitor and control all the exchanges happening between the Query components, we created a Query Interceptor component. It's main job is to capture all the queries transiting on the exchange channel, display them on a graphical interface to the users, and allow the users to forward them manually afterwards. This component serves two purposes: It enables demonstrating the reasoning process using a single graph that is created in a step by step fashion; it also facilitates debugging the system by centralizing all the exchanges in one component.

The Graphical interface of the Query Interceptor has two parts: the graph on the left that is used to visualize the information exchange; and the user controls on the right. When a query is sent from a component a to a component b through the interceptor, two vertices, a and b, and an edge from a to b are added to the graph. If the Interceptor is set to the *demo-mode* (by selecting a check box called "demo mode" on the Graphical User Interface GUI), the query is paused at the interceptor, and the user has to click the *Next* button in order to actually forward the query from the interceptor to component b. The same happens when a response is sent from b to a. If the *demo-mode* is not selected, all the queries and responses are forwarded automatically without any intervention of the user as if the Interceptor is transparent or turned off. The Interceptor also contains *Reset* button, which clears the graph and restarts the monitoring.

On the Kevoree platform, the Query Interceptor component has two ports: *Query In* port that receives all the queries sent from the Query components, and a *Query out* port to forward the query back to the components after being displayed on the Interceptor GUI.

7.3.5 Query class and loop detection mechanism

The query Java class that we developed for R-CoRe has the following attributes: the queried literal, the name of the component that initiated the query (*query owner*), the name of the component to which the query is addressed (*query recipient*), the id of the query servant thread that is responsible for evaluating the query, a set of supportive sets (set of foreign literals that are used for the evaluation of a query), and a list that represents the history of the query. The history is used to track back to the origin of the query by a loop detection mechanism, which we have integrated in the query evaluation algorithm.

As the query evaluation algorithm is distributed, we cannot know a-priori whether a query will initiate an infinite loop of messages. The loop detection mechanism that we developed detects and terminates any infinite loops. The simple case is when a literal $(c_i : a)$ in component C_i depends on literal $(c_k : b)$ of component C_k, and vice-versa. The loop detection mechanism works as follows: each time the query servant inquires about a foreign literal to solve the current query, it first checks that the foreign literal in question does not exist in the history of the current query, and if not, it generates a new query for the foreign

literal by integrating the history of the current query into the history of the new one. This way, a query servant is only allowed to inquire about new literals.

7.4 Demonstrating R-Core

Returning to our use case, the HCS receives data from different devices, and many situations may occur causing the data to be erroneous, e.g., Annette may have left her health bracelet next to her bed instead of wearing it, or the battery capability may be weak and preventing the bracelet from transmitting any data.

On the other hand, CDL allows to manage uncertainty and reason about it. The problem remains to apply such theoretical tools to the AAL domain in order to solve the very concrete challenges affecting patients. In this section, we present the Kevoree environment, which we use to address such issues by implementing the CDL reasoning model. This is illustrated in figure 7.4.

Figure 7.4: Kevoree bridges the AAL needs to the theoretical model of CDL.

7.4.1 Setup

Applying the above methodology on the running example described in section 7.2, we created 5 Query Component instances, each one representing one of the devices or elements of the scenario: the sms module, the bracelet, the medical profile, the ARM and the Home Care System. According to the scenario, the sms module must determine whether to send messages to the neighbors according to a predefined set of rules. Using a console component of Kevoree that we attached to the sms module, we are able now to initiate queries on the sms module.

Figure 7.5 shows our experimental setup, which involves the 5 query components, the console connected to the sms module (*FakeConsole*) and the Query Interceptor component. Note that all query input and output ports of the query components are connected to the Interceptor in order to allow us to capture all the exchanges for our demo session.

Before pushing the model from the Kevoree editor to the Kevoree runtime (i.e.: the node that will host the instances), we setup the properties of the components to initialize their knowledge bases and preference orders as described in table 7.1. For instance, the *sms* component is initiated with a knowledge base containing one mapping rule ($M1$) that states that if ($hcs : emergency$) of *hcs* is true, then ($sms : dispatchSMS$) of the sms module will also be true. HCSPref.txt contains the preference order of *hcs*, according to which the information imported by the medical profile is preferred to that coming from the ARM, which is in turn preferred to that coming from the bracelet.

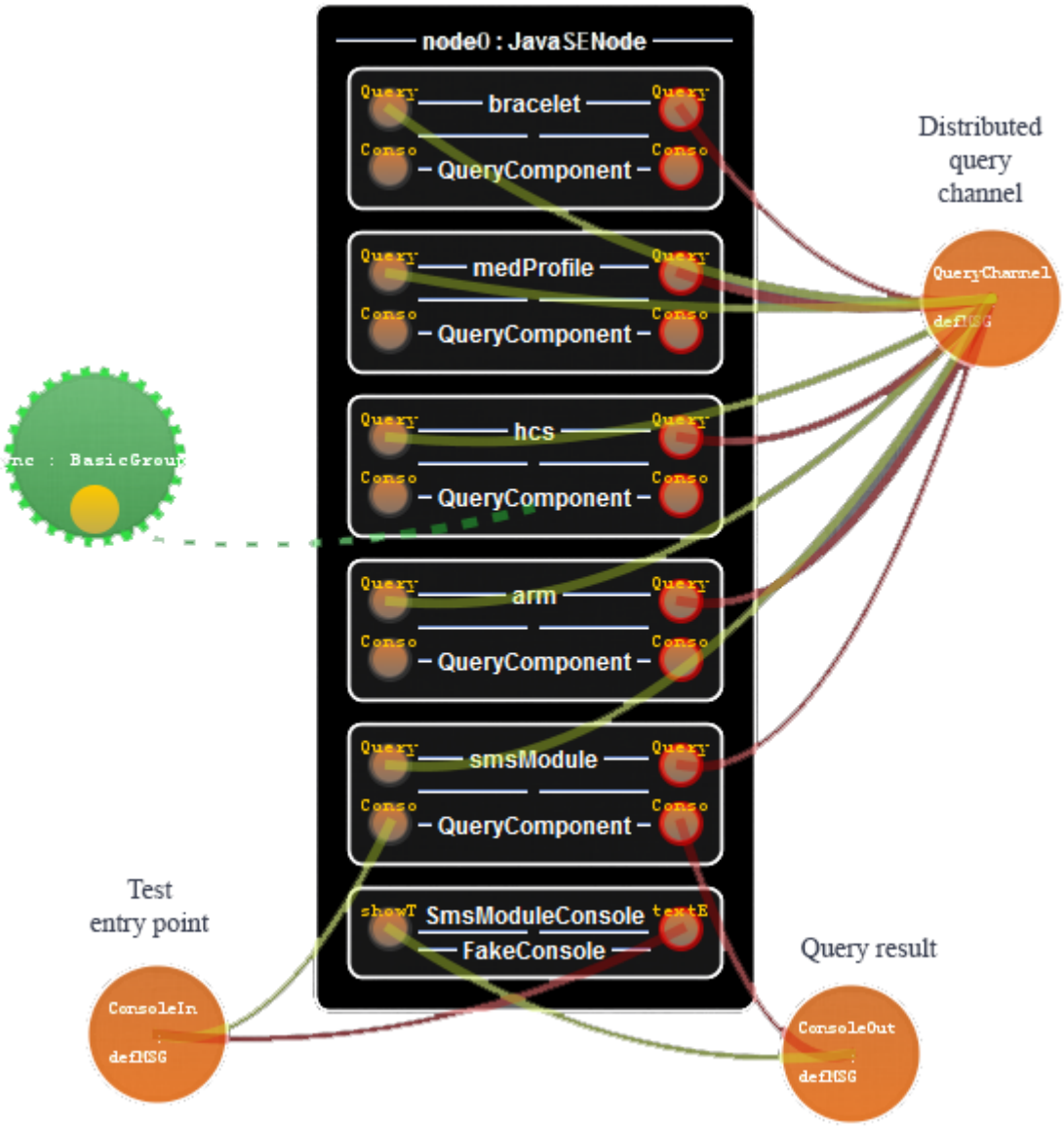

Figure 7.5: The running example implemented, a snapshot of the Kevoree Editor.

Table 7.1: Initialization of the components of the running example

File Name	File contents
SMSModuleKB.txt	M1: (hcs:emergency) → (sms:dispatchSMS)
BraceletKB.txt	L1: → (br:normalPulse)
MedProfileKB.txt	L1: → (med:proneToHA)
ArmKB.txt	L1: → (arm:lyingOnFloor)
HCSKB.txt	M1: (br:normalPulse) ⇒ ¬(hcs:emergency) M2: (arm:lyingOnFloor), (med:proneToHA) ⇒ (has:emergency)
HCSPref.txt	med, arm, br

7.4.2 Execution

After pushing the model to the Kevoree runtime, a console appears allowing us to interact with the SMS module. We initiate a query about $(sms : dispatchSMS)$ on the console by typing ***dispatchSMS*** on the console of the SMSModule.[ii]

The SMS module starts a new query servant which initiates in its turn a new query about $(hcs : emergency)$. This query is captured by the interceptor and it is displayed on the graph GUI; in fact two nodes representing the SMSModule and HCS are added to the graph. If the *demo mode* of the interceptor is selected, the user has to click on *Next* button each time to forward a query from one component to another. This step-by-step mode is very useful to understand the actual interactions and to slow down the exchanges.

The knowledge base of *hcs* contains one rule supporting $(hcs : emergency)$, $M2$, and another one supporting its negation, $M1$. *hcs* evaluates both rules and resolves the conflict using its preference order. Finally, it sends back the result of the query to the first query servant, which in turn computes and returns a positive truth value for $(sms : dispatchSMS)$. Each time a query or a query response is generated from any component and dispatched to any other, the interceptor captures it, adds it to the graph, and waits for the user to click Next before forwarding the query or the response to the appropriate component.

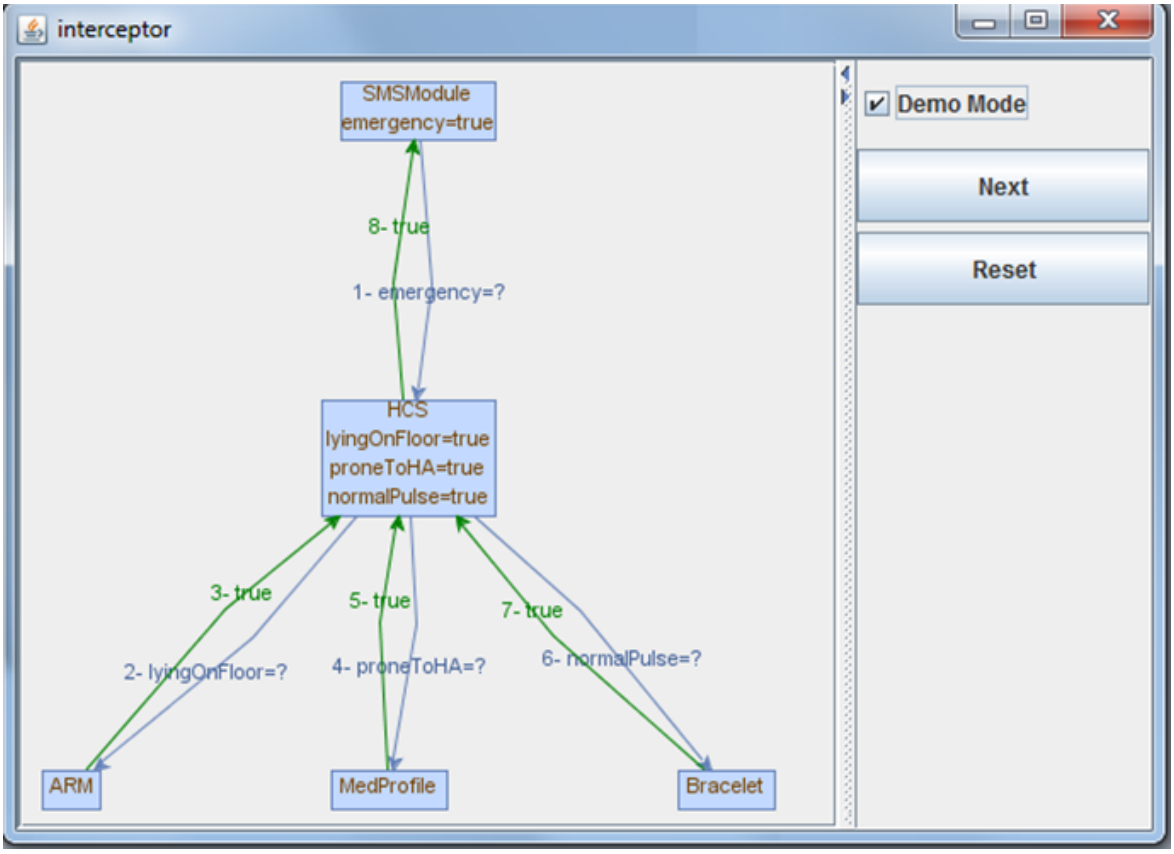

Figure 7.6: Different steps of execution of the running example.

[ii] You can run the demo and access its source code and all other necessary files at: `https://github.com/securityandtrust/ruleml13`.

Screen-shots from the demo during different steps of execution are displayed in figure 7.6. Following this, the SMS module will display, on the console connected to smsModule, the answer for $dispatchSMS$, which in this case is $true$.

Before pushing the model from the Kevoree editor to the Kevoree runtime (i.e.: the node that will host the instances), we setup the properties of the components to initialize their knowledge bases and preference orders as described in table 7.1, and according to the CDL representation model that we present in Section 2.5. For instance, the *sms* component is initiated with a knowledge base containing one mapping rule ($M1$) that states that if ($hcs : emergency$) of *hcs* is true, then ($sms : dispatchSMS$) of the sms module will also be true. HCSPref.txt contains the preference order of *hcs*, according to which the information imported by the medical profile is preferred to that coming from the ARM, which is in turn preferred to that coming from the bracelet.

After pushing the model to the Kevoree runtime, a console appears allowing us to interact with the sms module. We initiate a query about ($sms : dispatchSMS$), and we get $true$ as a response. In fact, what happens in the back-end is that a query servant starts on the sms module to solve the query. The query servant initiates, then, a new query for ($hcs : emergency$). In the knowledge base of *hcs*, there is one rule supporting this literal, and another one supporting its negation. *hcs* evaluates both rules and resolves the conflict using its preference order. Finally, it sends back the result of the query to the first query servant, which in turn computes and returns a positive truth value for ($sms : dispatchSMS$). The full interaction is displayed in figure 7.7.

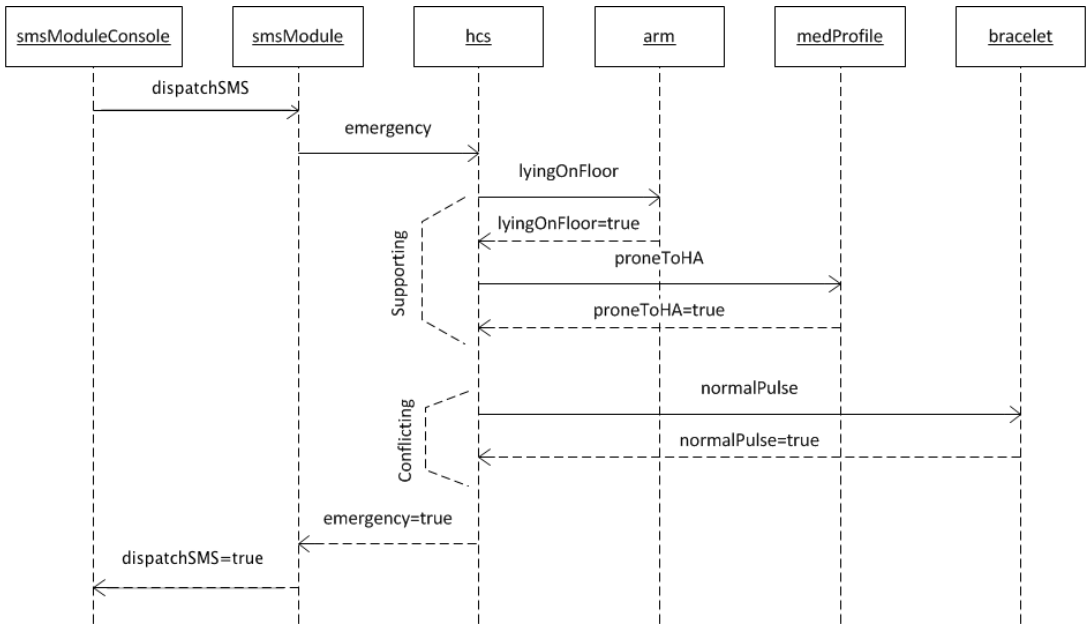

Figure 7.7: Execution of the running example.

7.5 Conclusion

We presented R-CoRe: a rule-based contextual reasoning framework for ambient intelligence and demonstrated its use ambient assisted living. Being based on contextual defeasible

logic, R-CoRe enables reasoning with imperfect context data in a distributed fashion. The capabilities of the underlying software platform, Kevoree, and its component-based models, enable R-CoRe to overcome several technical issues related to communication, information exchange and detection. R-CoRe, as currently presented in this chapter, has some technical limitations. As it deals with real components, we must assume limited memory, battery, computation and power resources. These limitations vary widely from a component to another depending on the nature of the components, and their technical complexity. For the current implementation, we have limited the knowledge base size to a maximum of 500 literals and rules. We have also limited the time-out for 10 seconds, so that if a component does not receive an answer to its query within 10 seconds, the corresponding thread server will send a time-out response, and the query will automatically expire. These limits define the maximum number of hops that a query can make before it expires, which in turn limits the communication resources, as some communication channels might not be free (over sms for example). With the current settings, we can easily implement small to medium scale AAL scenarios. However, dealing with more complex scenarios require including an optimization engine. To address such needs, the next chapters will discuss on solutions that offer trade-offs between computation time, memory and communication between devices. These optimizations techniques, when combined with models@run.time, will allow the AmI platform to adapt itself depending on the available resources, especially for large scale applications.

In this chapter, we enabled agents to intercommunicate and exchange information in order to derive the context in a distributed way. With this information in hand, the next step is to investigate how to adapt AmI platforms according to the contextual information.

8

Polymer - A model-driven approach for MOEA

After identifying the context in AmI, the system needs to execute an adaptation process accordingly. Most of the times, the adaptation involves a multi-objective optimization search to find a trade-off between the conflicting objectives. In the case of AmI, the objectives can be the different system qualities (privacy, utility and efficiency). Moreover, for a large scale deployment, the scheduling of tasks on the available resources can be as well converted to an MOEA problem. However, the current available MOEA frameworks require a specific genetic encoding of the problem. In this chapter, we present Polymer, a model-driven approach for MOEA that allows us to run optimization problems directly on domain-specific models without the need to do the encoding and decoding steps. Moreover, the contribution of this chapter is generic and not limited to AmI or IoT domains. It can be useful in many domains (including: finance, science, engineering). In the following chapter, we apply Polymer for AmI.

This chapter is based on the work that has been published in the following paper:

- Assaad Moawad, Thomas Hartmann, François Fouquet, Grégory Nain, Jacques Klein, and Johann Bourcier. Polymer: A model-driven approach for simpler, safer, and evolutive multi-objective optimization development. In *MODELSWARD 2015-3rd International Conference on Model-Driven Engineering and Software Development*, pages 286–293. SCITEPRESS, 2015

Contents

8.1 Introduction

In many domains, such as finance, science, engineering, and logistics several conflicting objectives need to be simultaneously optimized. In finance for example, the anticipated value of a portfolio should be maximized, whereas the expected risk should be minimized. Such problems, involving conflicting objectives, are called *multi-objective optimization (MOO) problems*, characterized by solutions offering trade-offs between different objectives.

Different algorithms can cope with such problems, *e.g.* particular swarm optimization [KE+95], simulated annealing [VLA87], and population based algorithms [DPAM02]. Multi-objective evolutionary algorithms (MOEAs) are another class of algorithms, which has proved to be particularly capable of finding solutions for complex domain-specific optimization problems with large solution spaces for which typically no efficient deterministic algorithms exist. However, MOEAs are difficult to use, require specific and detailed expert knowledge about fitness functions, mutation operators, and genetic problem encodings. Common encodings consist in mapping a domain-specific MOO problem into a binary, permutation-based matrix, or graph-based representation [CVVL02]. The solutions found by MOEAs must then be decoded, meaning to map them back to the domain-specific multi-objective optimization problem. This makes it very challenging to properly apply MOEAs and may require developers to focus more on the encoding of a problem than on the problem itself [KCS06]. The continuous design process of today's software systems makes it even more difficult to implement and especially to maintain MOEAs. Each change in the domain and/or in the optimization goals requires to adapt the problem encoding and decoding. Since the encoding is usually not statically typed, type checkers cannot be used to indicate potential errors. As a consequence it is hardly possible to use standard refactoring techniques to apply domain and/or optimization goal changes. Instead, the problem encoding must be adapted manually and independently from the domain representation. Moreover, optimization problems are inevitably linked to the growing complexity of software. Fitness functions and mutation operators can use these models and their API instead of relying on complex and error prone encoding steps. Similar to paradigms like domain-driven design, our model-driven approach allows developers to focus on the actual domain-specific optimization problems rather than on technical encoding details.

In this chapter, we present a new MDE approach to develop MOO layers directly on top of models. **This approach is generic and can be applied to any domain-specific model. In the next chapter, we will apply it for the AmI domain and show how it contributes to the big picture of the thesis**. Our approach also reduces the gap between MOEA representations and models, allowing to reuse standard modeling tools (*e.g.* model checkers) within fitness functions or mutation operators. Integrated into an open-source framework, our approach can significantly simplify domain-specific MOO development. We evaluate our approach on a cloud case study and show its suitability in terms of *i)* complexity to implement a MOO problem, *ii)* complexity to adapt (maintain) this implementation caused by changes in the domain model and/or optimization goals, and *iii)* efficiency and effectiveness remains comparable to traditional implementations.

8.2 Real-World Case Study

MOEA frameworks are often evaluated against a suite of mathematical functions with well known characteristics like ZDT, DTLZ and LZ09 [ABBZ12]. These problems are well suited to evaluate the performance of MOEAs in terms of finding the best Pareto front and the optimum solution. However, they are less suitable to represent the complexity of real-world problems and to evaluate the required effort for developers to implement and maintain MOOs for domain-specific problems. In this section we define a non trivial real-world MOO case study to apply our approach to. As shown in [FFH] the cloud computing domain is an appropriate real-world case study where MOEAs are used for scheduling and scaling tasks.

Scheduling applications on the cloud is a non trivial task [PWGB10]. First of all, several different cloud computing providers exist on the market. Each of these offers different computing specifications (*e.g.* CPU, memory, storage, and networking capacity) at different prices and pricing models (fixed pricing, bidding, etc). Then, different applications can be deployed on the cloud, each having different requirements in terms of hardware (CPU, RAM, disk, network), security (such as sand-boxing, redundancy), priority, and latency (distribution on different geographic zones). All of these requirements can be translated into optimization fitness functions. Moreover, a similar case study can be defined in the AmI domain. For example in order to better distribute computation loads among the available devices at a large scale, taking into consideration their cpu power and battery charge state.

For this chapter, we define a generic case study from the cloud computing domain: we have a fixed number $n = 7$ of applications, each requiring a specific computational power (in virtual CPU hours). Further, we assume that these applications can be parallelized and distributed into smaller tasks. Then we can rent a variable number $m < 100$ of cloud instances from a provided list of available instance models $M = 47$ (we provide as input for each: number of virtual CPUs and fixed price per hour). An instance model can be rented several times. We divide each application into m tasks and we assign a weight to each task (between 0 and 100). The weight w_{ij} of application i on instance j represents the proportion of resources allocated for the application i on the instance j. The optimization objectives are, *i)* reducing the average time required to execute all applications and *ii)* reducing the total price paid on instances. These two objectives are conflicting: the more we pay, the more instances we can rent and the less average time is needed for the execution. In addition, we define three mutation operators for this case study: *AddInstance* (to allocate a new cloud instance), *RemoveInstance* (to delete a cloud instance), and *ChangeWeight* (to randomly change the weight of an application on an instance). We implement the same problem with the same input files with our approach, using models and in a traditional way, using an array encoding with the jMetal framework [DN11]. jMetal is an object-oriented Java-based framework for MOO with metaheuristics. We use for both implementations a $population\_size = 20$ and $max\_generations = 1000$.

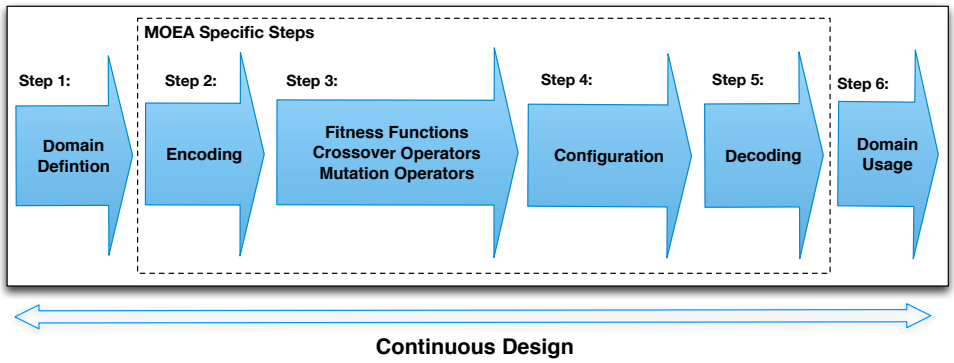

Figure 8.1: Classical steps to use MOEAs

8.3 Model-based MOO

In this section we describe our model-driven approach to simplify complex multi-objective optimizations.

8.3.1 Approach

The classical process of using MOEAs in an application can be divided into six steps, which are shown in figure 8.1. *First*, the problem domain must be defined and implemented. This is usually achieved using tools such as UML or E/R diagrams and standard programming languages like Java or C/C++ (POJO based). *The second step* is the encoding step. This means that the section of the domain impacted by the MOO must be mapped to a suitable structure in order to execute MOEAs. Typically, representations like binary or int arrays, permutations, matrix, or graphs are used for this purpose [CVVL02]. Figure 8.2 shows an example of such a classical encoding (based on an int array), aligned to our cloud case study. As for our case study, the number of reserved instances m can vary and is subject to

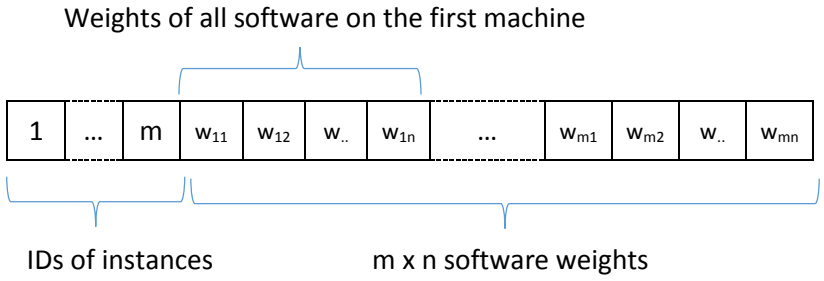

Figure 8.2: Classical MOO encoding using arrays

optimization, the size of the array varies as well. For each computing instance, we need to allocate an integer for the ID of the instance (in our case a number from 1 to 47), and n integers for the weights of the n applications. In this way the encoding consists of an integer array of size: $m \times (n + 1)$. As a design choice, we decide that the first m values

in the encoding represent the IDs of the reserved instances and the next $m \times n$ values represent the weights of the applications. However, this classical mapping comes with a number of drawbacks. First, this task is usually not trivial and requires expertise form both sides, domain experts as well as MOEA experts. Second, type-safety is lost. Consequently, it is more difficult to find bugs in the encoded MOEA representation and to maintain it. Last but not least, such encodings are difficult to read. *In a third step* fitness functions, crossover, and mutation operators must be defined. Listing 3 shows as an example, again aligned to our cloud case study, the implementation of a mutator to remove an instance.

Algorithm 3 RemoveInstance mutator on array encoding

```
class RemoveInstance implements Mutation {
void mutate(int[] cloud) {
int m= cloud.length/(N_SOFT+1);//number of instances
if(m==0) return;
int x= rand.nextInt(m);
int[] newCloud= new int[(m−1)∗(N_SOFT+1)];
for(int i=0;i<x;i++) { newCloud[i]=cloud[i] };
for(int i=x+1;i<m+x∗N_SOFT;i++){newCloud[i−1]=cloud[i]};
for(int i=m+(x+1)∗N_SOFT;i<m∗(N_SOFT+1);i++){
newCloud[i−1−N_SOFT]=cloud[i]}; cloud=newCloud; }}
```

As can be seen in the listing, in order to remove a machine x from the cloud, we need to remove its ID from the ID section then remove the n associated application *weights* from the *weights* section. *Next*, the MOEA setup must be configured. This includes the initialization of the population, its size, and the selection of an algorithm for the diversification of solutions (*e.g.* NSGA-II). *In a fifth step* the solution found by a MOEA must be interpreted and decoded back to the domain representation. *Finally*, the decoded solutions can be used in the domain-specific MOO problem.

A major challenge that comes along with this classical process is that in case of refactoring, and continuous design, all these steps have to be checked for impacts and potentially must be repeated.

We claim that this process can be significantly simplified by leveraging model-driven engineering techniques to allow a domain-specific model encoding for MOO problems. The main idea of this approach is to allow to express the MOO problem, fitness functions, crossover and mutations operators directly and seamlessly using the domain model, making explicit encoding and decoding steps unnecessary. This allows to throughout use the same models for both the domain representation and MOO problem encoding and therefore to avoid this mismatch. By integrating our approach into the open-source *Kevoree Modeling Framework (KMF)* [FNM+12b] we provide a framework[i] to express domain-specific model encodings for MOEAs in order to discard the encoding/decoding steps. Figure 8.3 shows an overview of the layers involved in our approach. In MDE the application layer is typically built using a modeling API, which is generated form a meta-model definition (*e.g.* from an Ecore model). In our approach, we use MOF based modeling, using textual and graphical representations. We extend this modeling API with several interfaces to implement fitness

[i] http://kevoree.org/polymer/

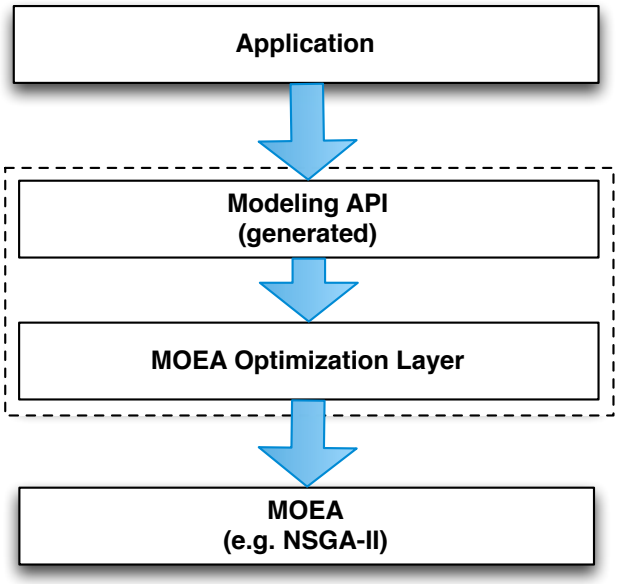

Figure 8.3: Model-based MOO overview

functions, mutation and crossover operators, as well as the population creation factory. These interfaces are typed with the model, which means that they can be implemented using the modeling API. Listing 4 shows a simplified version of our interfaces.

Algorithm 4 Interfaces of the extended modeling API

```
interface FitnessFunction<A extends KContainer> {
double evaluate(A model);}
interface MutationOperator<A extends KContainer> {
List<MutationVars> enumerateVars(A model);
void mutate(A model);}
interface CrossoverOperator<A extends KContainer> {
A execute(A modelA, A modelB);}
```

Figure 8.4 represents the model encoding of the MOO problem of our cloud case study. As can be seen in the figure, a cloud contains several virtual machine instances and

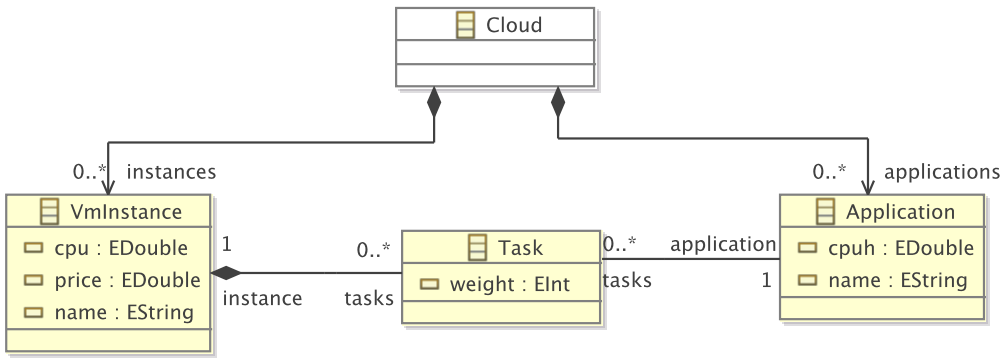

Figure 8.4: A cloud meta-model

applications, which can be executed on the virtual machine instances. Using standard model-driven techniques this meta-model can be transformed into object-oriented code. Listing 5 again shows the implementation of the *RemoveInstance* mutator. But this time the operator is implemented directly on the model encoding instead of an explicit MOO problem encoding.

Algorithm 5 RemoveInstance mutator on a model encoding

```
class RemoveInstance implements MutationOperator<Cloud> {
void mutate(Cloud could) {
int m=could.getInstances().size();
if(m==0) return;
int x= rand.nextInt(m);
VmInstance vmtoRemove = could.getInstances.get(x);
could.removeInstances(vmtoRemove); }}
```

The extended modeling API allows developers to encode the MOO problem completely in terms of the domain model. In addition, we provide a MOEA optimization layer, which initiates the population, calculates the fitnesses, and executes mutation and crossover operators according to their implementations. We use established algorithms such as NSGA-II and ε-MOEA for the actual MOO. These algorithms only operate on the provided fitnesses and are independent from our model encoding. Our optimization layer provides the fitnesses and executes the mutations (both on top of models) for the underlaying MOEA layer. The MOEA optimization layer can be configured to choose one of the supported MOEAs and to set the maximum number of generations.

This approach enables the execution of MOEAs on top of models and can significantly simplify the usage of MOEAs to tackle domain-specific MOO problems. The necessary MOEA expertise is hidden in the framework and allows software engineers to focus on the domain-specific MOO problem, instead of MOEA encoding and decoding. First, software engineers can express the optimization problems in terms of the domain model without being MOEA experts. Second, type-safety is maintained, which improves the maintainability of the application. Third, since the problem is expressed in domain terms the solution is far more readable. Leveraging type safety and object-oriented design, MOEAs can shift to complex optimization using several mutation operators. Last but not least, our approach enables to use standard modeling techniques during the optimization process, and thus can leverage many tools to verify solutions.

8.3.2 Polymer Implementation

We implement our framework on top of KMF [FNM+12b], which is an alternative to the Eclipse Modeling Framework (EMF), fully supporting the Ecore file format but targeting runtime performance. We use KMF to generate the modeling code and API from domain models to support the construction of MOEA elements (fitness functions and mutation operators) on top. The Polymer framework core itself on one hand provides the MOEA interfaces described in section 8.3.1 and on the other hand implements a mapping layer to

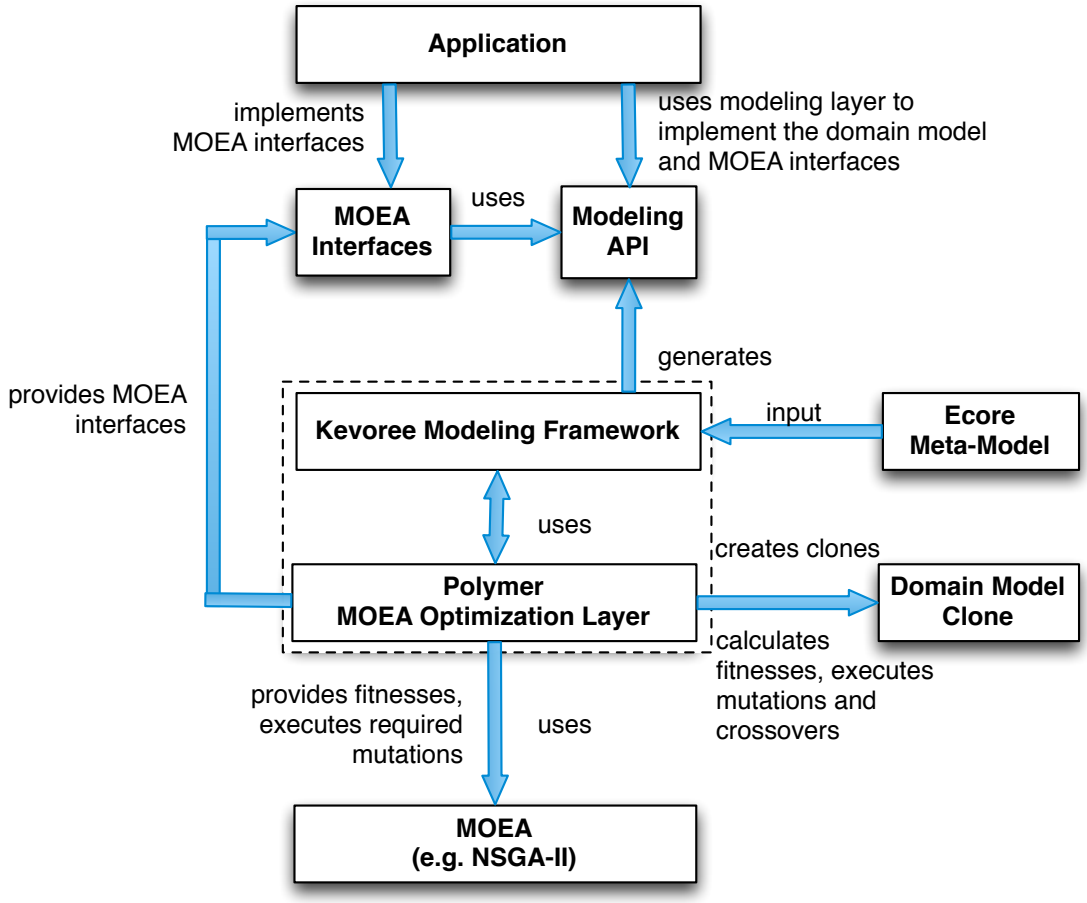

Figure 8.5: Model-based MOO implementation

intermediate between classical MOEAs and our model-based encoding. Therefore, Polymer calculates the fitness function score based on the implementation of the provided interfaces, using a classical visitor pattern. The result of the different fitness functions (plain numbers) are used as input for classical MOEA Pareto-front selectors. Whenever a MOEA needs to mutate one of the solutions, Polymer creates a clone of the domain model and executes one of the implemented mutation operators (randomly) on this clone. Since cloning of models is a frequent but costly operation in our approach we implemented an efficient partial cloning mechanism, which is described in more detail in section 8.3.3. Like fitness functions, mutation operators operate directly on the generate modeling API and can take advantage of model transformations. In this contribution, we do not intent to provide a new MOEA algorithm, instead we allow to use different already existing search algorithms. The novelty of our approach is to allow the usage of a model-based encoding for domain specific MOO problems. Figure 8.5 summerizes the implementation concept of Polymer.

8.3.3 Partial Model Cloning

It is easy to efficiently copy traditional MOEA problem encodings, *e.g.* binary arrays. However, our model-based problem encodings are more difficult and less efficient to copy, since potentially a complete model must be cloned. To reach a comparable efficiency in terms of memory usage and performance, while still offering developers the advantages of working directly on top of domain models, we provide a partial model cloning mechanism [FNM+14]. First, for most real-world applications, we argue that only a fraction of parameters and fields of domain-specific models are subject to optimization. A big part of the model consists of immutable fields or reflects static characteristics of the domain and consequently is outside the search space of MOEAs. Therefore, whenever we clone a model to generate another potential solution, we ensure in our framework to not copy immutable fields but only refer to them. In fact, this mechanism only clones the mutable parts of a model (partial cloning) instead of the whole model. For our case study, the application class can be declared as immutable, because the mutations do not impact this class. In the cloud case study, the optimization occurs on the rented machine and the way the weights are distributed. This partial cloning is completely transparent for developers.

8.4 Evaluation

We evaluate our approach on the cloud case study and show its suitability in terms of *i)* complexity to implement a MOO problem, *ii)* complexity to adapt (maintain) this implementation, and *iii)* show that the efficiency and effectiveness of our approach remains comparable to ad-hoc implementations.

8.4.1 Complexity to Implement

We claim that a major complexity of MOEA based developments is due to the required mixed expertise, on the one hand MOEA and on the other hand the domain itself. To evaluate the complexity of our approach compared to classical MOEA development we implement the cloud case study in both approaches and count the required lines of code (LoC) associated for each expertise. We keep the same coding style for both implementations in order to keep the absolute number of LoC comparable, however the most important point is the balance between MOEA and domain expertise. The Polymer implementation takes 158 LoC, while the implementation with jMetal needs 189, as shown in table 8.1. This difference is mainly due to the encoding/decoding in the jMetal implementation. Additionally, it is important to notice that the 24 LoC for the domain definition of the Polymer version are reusable for other model-based developments. For jMetal, in every operator, the genetic encoding must be defined by MOEA experts, in order to be usable by domain experts. That explains the 71 LoC difference for the MOEA experts. For Polymer, MOEA experts have just to decide the core settings (the algorithm and its configuration, number of generations, number of individuals in the population). However, the 40 additional lines difference for the domain

Feature	jMetal		Polymer	
	M. LoC	D. LoC	M. LoC	D. LoC
Domain Encoding	1	–	–	24
Fitness Operators	13	39	–	39
Mutation Operators	28	32	–	34
Crossover Operators	6	22	–	22
Population Factory	6	16	-	30
MOEA settings	14	–	9	–
Decoding solution	12	–	–	–
Total	80	109	9	149

Table 8.1: Implementation complexity for jMetal / Polymer. M.Loc=MOEA Expert LoC / D.Loc=Domain Expert LoC

experts in the Polymer implementation are due to the model structure and manipulations of objects where in the traditional implementation the structure is fixed in the array encoding. We can conclude from this subsection that the Polymer based implementation successes to reduce the development complexity by allowing developers to focus on cloud optimization (the actual domain problem).

8.4.2 Evolutive Refactoring Robustness

In order to evaluate the necessary refactoring effort for both platforms, we consider 10 different modifications (in 3 categories) on the previously defined case study. We then count the modified lines of code to adapt the implementation for both frameworks. Additionally, we check if the necessary changes can be pointed out by the Java type checker or not.

In the first category, we remain in the same optimization problem definition, but consider adding/removing fitness functions, mutation, and crossover operators, over the same problem. Both, Polymer and jMetal are able to check the type of the operators.

In the second category, we remain again in the same optimization use case but we change some parameters, *e.g.* adding/removing an application. In jMetal it is necessary to manually add/remove this application from the array of applications in each crossover/mutation operator and fitness function. In Polymer, we just need to add/remove this application from the population creation factory. We do not need to modify any operator or fitness because the model is integrally passed to them. Another modification that falls into this category, is to add an optimization field. Let's say we want to optimize over CPU and network resources of the deployed applications on the virtual machines. In Polymer, we just have to add a field "network" in the metaclasses $Task$, $Application$ and $VmInstance$. There is no code change in any previously implemented fitness function, mutation and crossover operator necessary. However, in the traditional approach a big change needs to be **manually** done on the encoding. Figure 8.2 is not enough anymore to represent the new encoding. Instead, a new integer needs to be reserved for each application task on each VM instance. The total number of integers in the array representing a solution, changes from

the previously calculated $m \times (n+1)$ to the new value of $m \times (2n+1)$. Developers then need to manually update **all** previously defined operators in order to take this structural change in the encoding into account. In this simple use case implementation, we already counted as many as **43** lines of code affected by this change. What is even more dangerous, is that it is up to the encoding designer to define what comes first in the encoded array, among the v-cpu weight and the network weight. Type checkers cannot enforce that the encoding is actually used by its intended meaning in fitness functions and mutation operators. Table 8.2 summarizes the modifications.

Modification	**jMetal**		**Polymer**	
	LoC	T. Check	LoC	T. Check
Adding Mutator	3	✓	1	✓
Removing Mutator	3	✓	1	✓
Adding Crossover	3	✓	1	✓
Removing Crossover	3	✓	1	✓
Adding Fitness	1	✓	1	✓
Removing Fitness	1	✓	1	✓
Adding an App	10	✗	1	✓
Removing an App	10	✗	1	✓
Adding Network Optim.	43	✗	1	✓
Changing Optim. Problem	All	-	40%	-

Table 8.2: Lines of code changed in both frameworks

8.4.3 Performance and Effectiveness

To evaluate the effectiveness of our approach we run both implementations (model-based and traditional MOEAs) 100 times on our cloud case study on a core i7 computer (2.7Ghz) with 2 GB RAM dedicated for the optimization process. As shown in figure 8.6 both approaches lead to similar Pareto fronts. We validate this using a Mann-Whitney U-Test (statistical test at 0.01 significance level) which give evidence that both Pareto are comparable. At the core both frameworks use the same MOEA algorithm, namely NSGA-II. Therefore, we can conclude that the bias introduced by the modeling layer does not impact the result.

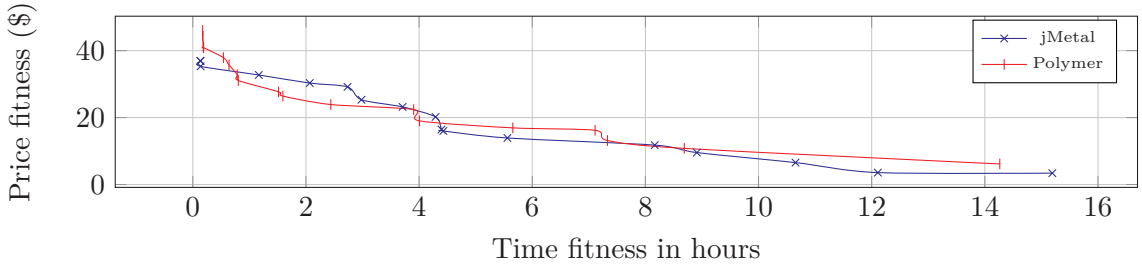

Figure 8.6: Pareto fronts for jMetal and Polymer

In order to evaluate the efficiency of our approach we measure the time for performing the solution search and the used memory. All in all, Polymer takes more time (by an average of 15 %) and memory (by an average of 20 %). This is due to model manipulations and cloning of models instead of simply copying arrays. The overhead in performance and memory is a trade-off to achieve an easier and more maintainable MOO development. However, this overhead decreases in term of percentage, when the complexity or the number of fitness functions to evaluate increase. For each genetic generation step the model cloning operation has a fixed cost. In order to optimize this step we apply the partial model cloning strategy. When dealing with complex fitness evaluations, the cloning cost and the overhead in performance become negligible (less than 5 % for the partial clone, and less than 15 % for the full clone). Figure 8.7 shows how the performance overhead (using as reference zero the traditional ad-hoc encoding) changes when the fitness functions become more complex to evaluate. As can be seen in figure 8.7 the overhead introduced by the partial cloning and model operations, become more and more negligible with increasing complexity of the case study. This increase in complexity can be due to one of three factors: 1) Increase in the number of objectives to optimize, *e.g.* optimizing network, RAM, disk usage, latency, security, redundancy and price at the same time. 2) Increase in complexity of previously defined fitnesses, *e.g.* a more precise fitness function is implemented. 3) Increase in the complexity of the domain itself, *e.g.* adding more virtual machines or more applications to optimize leads to slower fitness evaluation per generation. We can conclude, that our approach can compete with traditional MOEA usage by using heuristics like partial cloning. We can also see that the overhead becomes negligible for complex optimization scenarios, which are the main target of our approach (because for them maintainability becomes critical).

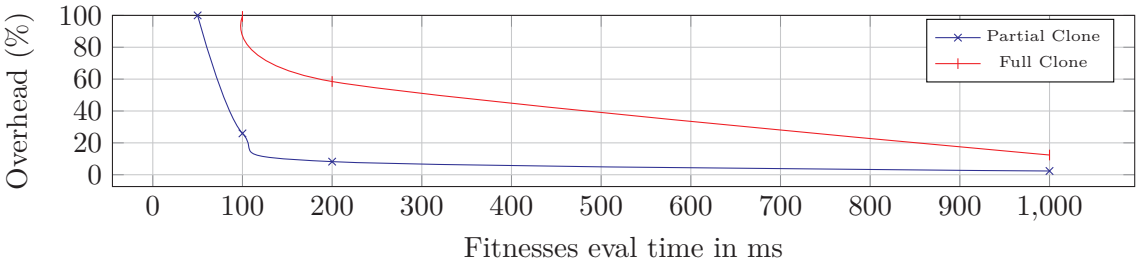

Figure 8.7: Performance overhead

8.5 Conclusion

We claim that the usage of MOEAs to solve domain-specific MOO problems is complicated. This is mainly due to necessary encoding steps, meaning that a domain-specific problem must be mapped into a structure, which is suitable for the execution of MOEAs. This burdens developers with not only to understand a domain-specific MOO problem but also with the technical challenge to properly express this problem in terms of MOEA

encoding. Many application domains can benefit from MOOs on top of models. In this chapter, we introduced a MDE framework to allow developers to combine MOOs with a model-driven development process. For this purpose, we enabled the execution of MOEAs on top of models and significantly simplified their usage and improved their reusability. We showed that the necessary MOEA expertise is hidden in the framework and enables software engineers to focus on domain-specific MOO problems instead of MOEA encoding and decoding. This has several advantages. First, MOO problems can be expressed in terms of domain models without being a MOEA expert. Second, type-safety is kept, which improves maintainability of the application. Third, since the problem is expressed in domain terms the solution is more readable. Our approach allows to use the same models for domain representation and MOO problem encoding to avoid the mismatch between these representations.

The contribution of this chapter is to enable MOEA execution on top of complex model structures. For the big picture of the thesis, we show in the next chapter,how we apply the polymer framework for the AmI domain, mainly to search for the best trade-off between the several system qualities according to the current context, directly on top of component based models.

9

Adaptive blurring to balance privacy and utility

After presenting a generic framework to run MOEA directly on domain specific model, in this chapter we show how such framework can be used in the AmI environment to adapt the system according to the current context. For every context, we take the privacy, utility and efficiency qualities as the objectives to be optimized. We introduce blurring components as our main privacy preservation elements. Furthermore, we show how by combining MOEA on top of architectural component-based models, we can address the adaptability at a near real time challenge for AmI systems.

This chapter is based on the work that has been published in the following paper:

- Assaad Moawad, Thomas Hartmann, François Fouquet, Jacques Klein, and Yves Le Traon. Adaptive blurring of sensor data to balance privacy and utility for ubiquitous services. In *SAC 2015-The 30th ACM/SIGAPP Symposium On Applied Computing*, pages 2271–2278. ACM, 2015

Contents

9.1 Introduction

Given the trend towards mobile computing, the next generation of intelligent ubiquitous services must be more and more able to process sensor data of surrounding environment to enable new services for end-users. More than ever, such services will rely on data potentially related to personal activities to perform their tasks, *e.g.* to predict local weather using in-house weather station sensors, or urban traffic conditions as nowadays trending applications like Waze [i]. From the perspective of service providers high quality data are supposed to be: *1)* accurate, *2)* fresh enough for the service, and *3)* diverse to reveal potential correlations. However, such data are sensitive and often related to personal activities and therefore can lead to privacy risks, *e.g. profiling* [MM09] and *inventorying* [Jue06], [Kel11]. For example, sharing the precise electric consumption of a house enables an analysis of the energetic performance, for example in order to enhance the heating system. However, recent results [MM09], [Kel11] demonstrate that sharing precise consumption data also leads to privacy issues. For instance, the currently watched TV program or inventory of a house can be inferred using physical signatures (electrical consumption, frequency, ...). Similarly an high precision position or in-house activity data gave a lot of information of user habits. This privacy problem can be generalized and applied to the forthcoming IoT domain, where sensors will be accessible to various stakeholders.

Despite a user wants to preserve his privacy as much as possible, he also needs to share enough data to allow services to work properly. The challenge is how to provide added-value services and preserve user privacy at the same time. Moreover, a user might install new services at any time and thus user's privacy configuration might change over time.

Classical access control systems, which only forbid or grant access (`'all or nothing'` access), can protect the access to personal data. Similarly, algorithms like K-anonymity can ensure the protection of sensor platform privacy but require that there is no relationship between sensor data and its origin. Such string assumption can forbid numerous smart services which will need at least a geographical zone origin to perform their computation. Usually this leads to situations where third-party services either can access all data or cannot access the data at all. In contrary, blurring techniques, by gradually decreasing data quality, offer a wide range of trade-offs between ultimate privacy (sharing nothing) and ultimate functionality for service providers (sharing everything with best quality). Blurring mechanisms allow to tune the quality of shared data for dedicated purposes, (*e.g.* by reducing the sampling rate of electric consumption to avoid foot-printing of plugged devices). Instead of having an `'all or nothing'` access to data, we suggest to use *blurring controlled data flows* between data producers and consumers to allow proportional data access control. Composed as a *chain*, blurring algorithms take raw data generated from sensors as input and produce a blurred stream as output. However, the amount of sensors and services can quickly lead to an explosion of possible configurations of blurring algorithms, which are hardly manageable for users. In order to tackle this complexity, we propose a generic

[i] http://waze.com

software platfom that automatically reconfigures and adapts (at runtime) the blurring chain between data owners and data consumers (services). Thus, this platform searches the optimal trade-off between service utility and privacy risks using multi-objective evolutionary algorithms in order to drive an underlying communication platform. This enhances user privacy by adapting blurring components according to a users privacy configuration and service requirements.

We aim to enable an ubiquitous sensor mediation platform in order to control data flows. In order to realize this, we combine the idea of blurring with concepts from dynamic, component based platforms, the M models@run.time paradigm, and techniques from search-based software engineering. More specifically:

- To control the "quality" of data to match a tolerated risk level, we use a set of dynamically deployable *blurring components*. By chaining these components between a data source (sensor) and a target (service) we can ensure the required privacy properties.

- We apply models@run.time techniques to use a model of the current running system as a reflection layer and to drive an underlying communication platform. We use the Kevoree [FBP+12] framework, which is an implementation of the models@run.time paradigm, in order to dynamically adapt blurring components at runtime.

- We leverage search-based software engineering methods (MOEAs) to search the best trade-off between service utility and privacy risks, in term of blurring components configuration, according to the tolerated risk level per service and context.

In this chapter, we apply the contribution we presented in chapter 8 for the AmI domain, in order to create an adaptable platform that finds a trade-off for the different qualities according to the current context, as presented in figure 9.1.

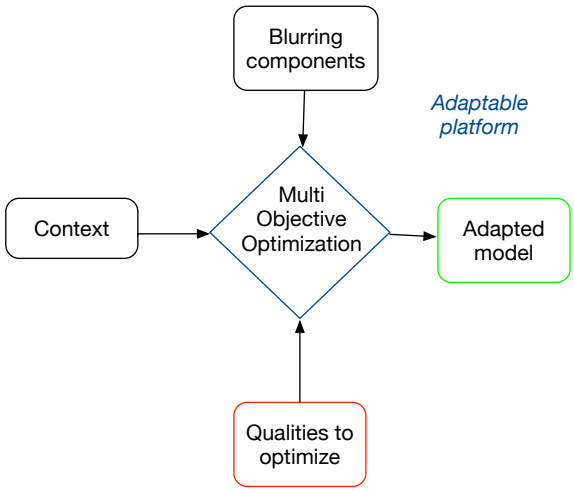

Figure 9.1: Adaptable platform

9.2 Proportional Data Access

Our approach aims at allowing users to control and limit privacy risks when sharing data collected from sensors. In order to make this possible, we use blurring components to dynamically adapt the quality of the shared data between real sensors and dedicated service *proxies* exposed to data consumers. In this chapter, we do not consider the security of the communication between the proxies and data consumers, nor the problem of authentication. We suppose that this communication channel is secured (*e.g.* via SSL) and an authentication mechanism to identify data consumers is implemented on the platform (*e.g.* HTTPS).

Instead, in this chapter we focus on the communication flow and the privacy risks, which occur when sharing high quality data. Thus, our research question is how to find an optimal chain of blurring components (and settings) between sensors and connected data consumers (*e.g.* identified as HTTP session) in order to balance user privacy and utility for ubiquitous services. Figure 9.2 shows an example of a blurring chain between a sensor x and a service proxy y.

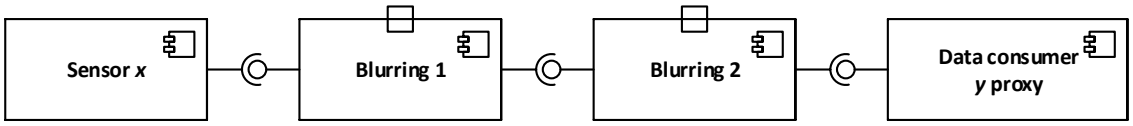

Figure 9.2: Chain of blurring components between a sensor and a data consumer.

Let's denote by X the total number of sensors available on a platform and by Y the total number of services (data consumers) currently connected to this platform. Each service wants to get data from a specific subset of sensors available on the platform. The data flow from a sensor $x \in X$ to a service $y \in Y$ introduces a privacy risk R_{xy}. To identify such privacy risks we assume the availably of a risk knowledge base, filled and maintained by security and privacy experts. Moreover, a sensor platform user (or owner) might not trust each data consumer to the same degree. Therefore, our approach foresees that a user can specify different privacy risk tolerances for different data consumer groups (*i.e.* family, friends, or strangers). This is inspired from the social network model of managing privacy [Mad12], which define groups of different trust levels. The amount to which a data flow between a sensor and a service should be blurred depends on these risk and trust levels.

In order to counter the privacy risks we provide in our platform different blurring algorithms. All counter strategies are indexed in a knowledge base, which we call *counter-measure database*. Every counter-measure (blurring strategy) is implemented in a dynamically deployable software component. The goal of our approach is to select the blurring component configuration (chain of blurring components) that best fulfills the privacy of users and the utility of the services. A third criteria that we include in the optimization is the performance of the dynamic communication platform. This means, besides the privacy and utility qualities we also take the resulting performance of the blurring components into account.

Given the fact that we map blurring algorithms to components, we reduce the problem to an architectural configuration. This reduces the problem of finding a trade-off between privacy and service utility to the problem of finding an appropriate chain of blurring components between sensors and connected devices. In order to select and configure such components we use an evolutionary algorithm to explore potential solutions. More precisely, we leverage a multi-objective evolutionary algorithm in order to find an acceptable solution, which balance the trade-off between privacy, service requirements, and platform performance.

Our framework is built around a reasoning engine, which reads the current state of the platform using the reflexion layer of models@run.time and optimizes it. In order to find alternatives our reasoning engine is fed with privacy risks, and counter-measures from our knowledge base (outside the scope here, for instance see the work of Neustaedter et al. [NGB03]). Figure 9.3 shows an overview of the proposed architecture. This architecture is based on an infinite loop between the runtime, which provides its current architecture model and the reasoner which searches and propagates the best alternative architecture to the runtime when necessary. Applying the models@run.time paradigm, this alternative solution can be deployed without interrupting the running system by updating only impacted elements.

To summarize, the proposed framework is build around a continuous adaptation loop,

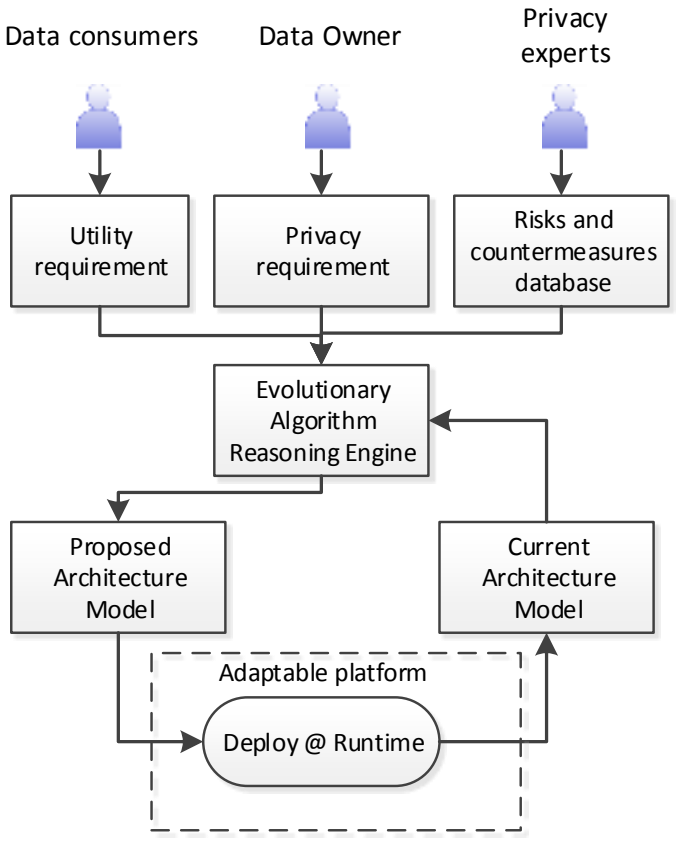

Figure 9.3: Adaptive blurring framework architecture.

which monitors and drives an underlying component platform through architecture models. Our framework takes as input *(i)* privacy requirements from users (data owners), *(ii)* utility

requirements (quality of data) from services (data consumers), *(iii)* risks and countermeasures provided by security experts.

Figure 9.4 shows an example of the state of the dynamic communication platform after a first deployment. Each service is in a different privacy group and had asked for different data sources, resulting in independent blurring chains. Our contribution is scoped between sensors and services in order to dynamically balance user privacy and utility qualities of connected ubiquitous services.

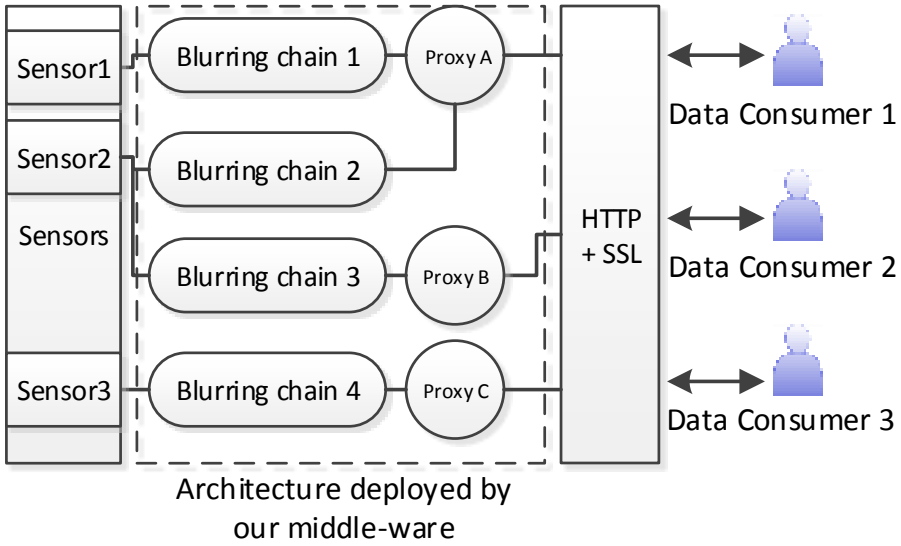

Figure 9.4: Example of a deployed architecture

9.3 Adaptive Blurring Platform

In this section, we describe the major elements of our proposed adaptive blurring framework.

9.3.1 Blurring Components

We propose to use *blurring components* in order to control the privacy risks that may occur when sharing high quality sensor data with data consumers. Under the term "blurring component" we understand a software component, which takes raw data as input and produces a blurred stream as output. The idea behind this approach is that, by gradually decreasing the data quality, a blurring component is able to hide some of the personal data delivered by sensors. However, at the same time, a blurring component still reveals enough information to allow services to work properly. For example, a service could automatically provide a user with the current weather of his whereabout. This service would need only the approximate location of the user to check the current weather in this area (typically a city). However, it is not necessary to share the exact location of the user with this service.

A blurring component could take the raw data from a sensor (exact location of the user), blur these data (approximate location), and provide the weather service only with the blurred data. This would allow the weather service to fulfill its task but, on the same time, increases the protection of personal data. The *blur intensity* can be parametrized to provide different blurring levels, ranging from directly sharing sensors' raw data without blurring to not sharing data at all. In order to blur data, different concrete blurring strategies can be applied on (raw) data. We provide two main categories of blurring components. The first category directly operates on *data values*, whereas the second category operates on the *time dimension* of data.

Time blurring components: We define three concrete blurring components in this category. The first one is *averaging*. This component takes the input data and calculates an average value (over a configurable time window). The blurring effect achieved by this component smooths and streamlines the output value. An example is the smart meter domain, where it is recommended to average the measurements over a window of 15-minutes [KFB11]. The second blurring component we provide in this category is *frequency reducing*. By limiting the number of output values (not all input values are used as output) per time window. This decreases the data quality by reducing the frequency. An use case example is the location privacy preservation domain: if the frequency of the data is high enough, an attacker can link different locations together and trace a full route of a single user [HGH+08]. *Access control* is the last blurring component we provide in this category. The idea behind is to restrict the access to the data during specific time periods. An example is the public video surveillance domain, where the video stream can be shared or not in certain periods of time [BTE+05].

Value blurring components: For this category we define three subcategories, *noise*, *pass filters*, and *generalize*. *Noise blurring* effects add specific properties and variances to the original data to make them less precise. These components are usually used for real-time applications, due to their fast performance. *Gaussian blurring* is a specific type of noise blurring. It uses a Gaussian distribution model to generate the noise [DKM+]. *Pass filters*, be it *above threshold* or *below threshold* filters, output the input data only if they pass above or under a certain threshold level or just use the boolean state when a value is above/under a defined threshold. These filters are useful, for instance, in the health care domain to share the data flow of sensors when data measurements go above or under certain medical critical limits. *Generalizing blurring* techniques reduce data accuracy, *e.g. trimming* rounds floating point numbers to a specific precision. Other generalizing techniques can include sending an interval or a range of answers instead of one value.

Blurring components can be domain independent (like a Gaussian noise generator) or domain dependent (such as location cloaking algorithms). We categorize these different blurring components using a type hierarchy meta-model. Figure 9.5 presents the different blurring components integrated in our platform.

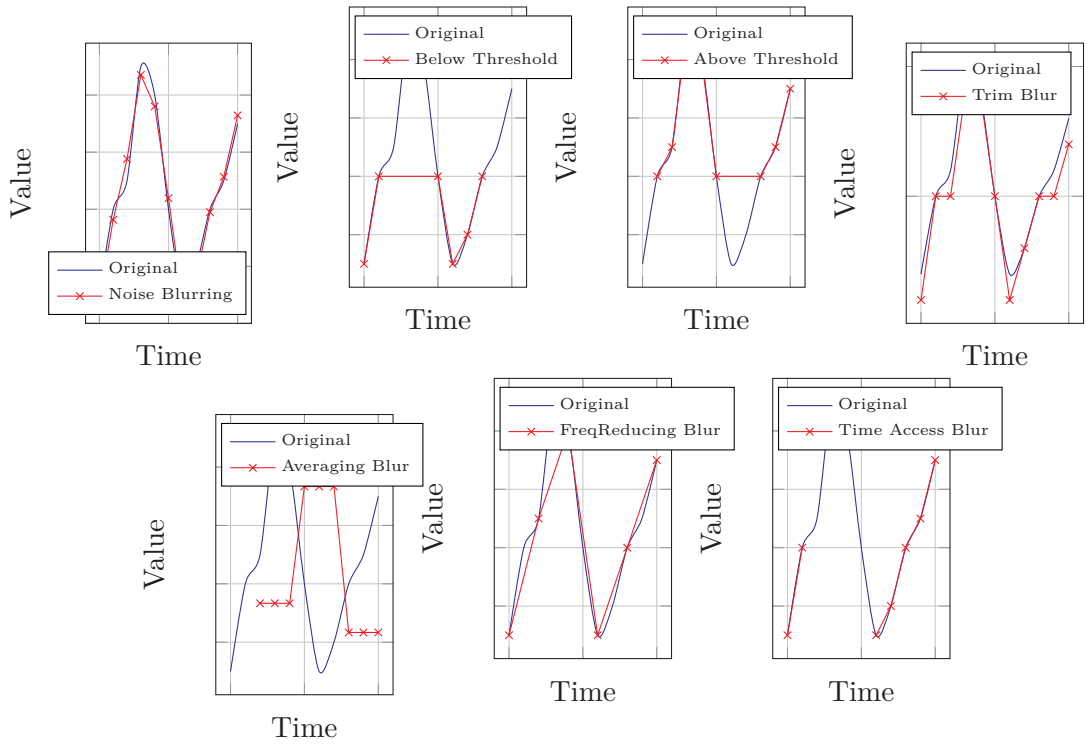

Figure 9.5: Different blurring techniques

9.3.2 Risk and Counter-Measure Model

In order to quantify privacy risks and counter-measures we use a knowledge base that has to be filled, maintained, and updated by domain privacy experts. For each sensor in the platform, these experts have to define a list of potential privacy risks that are related to that particular sensor and add a criticality weight for each of the risks. Then, different counter-measures for each defined risk have to be added.

A counter-measure for a risk consists of defining the blurring component and its risk reduction profile. For the sake of convenience and simplicity, we define the risk reduction profile by two mapping points and a mathematical profile. A mapping point links the blurring intensity to a risk reduction factor. The mathematical profile dictates the shape of the risk curve in between these two mapping points. In our implementation it can be linear, exponential, or logarithmic. In future work we plan to integrate more complex mathematical profiles to the platform.

For example, for the Gaussian noise blurring an expert can state that an intensity of 0 (variance of the noise=0) will have no impact on the risk in question (risk reduction factor=0) and an intensity of 3 (variance=3) is enough to remove 95% of the risk, thus a risk reduction factor of 0.95. Then, for example he can state that the risk reduction is linear between these two mapping points.

Figure 9.6 shows an simplified excerpt of the risk and counter-measure meta-model as implemented in our platform. The meta-model contains five main concepts: domain, sensor, risk, counter-measure, and blurring.

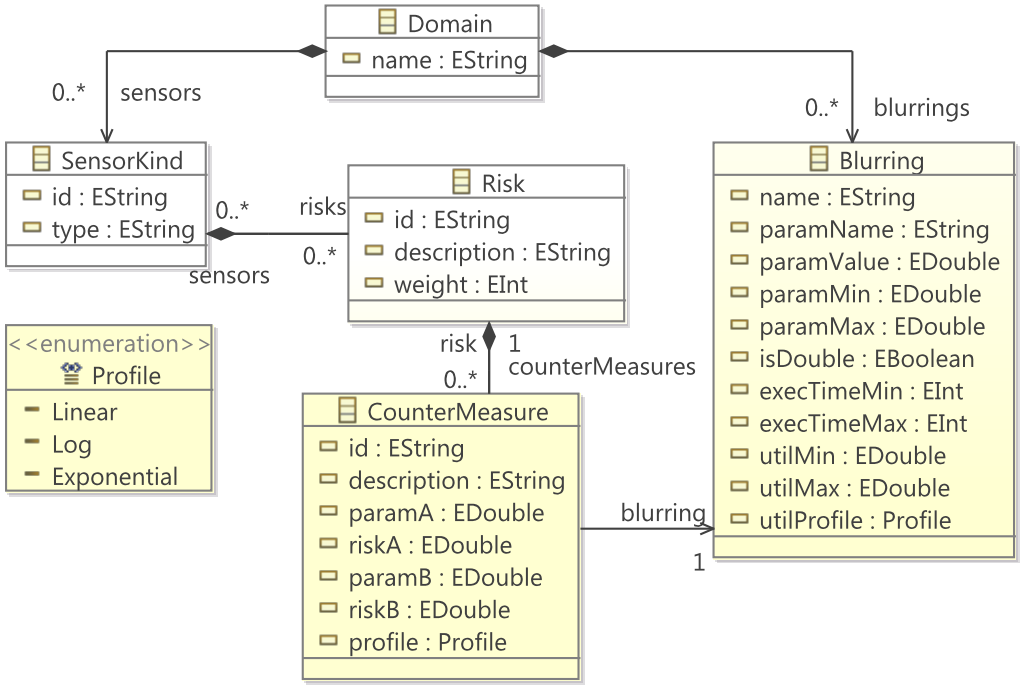

Figure 9.6: Risks and Counter-measures meta-model

Now, having this knowledge base the list of blurring components, and their intensity parameters we can derive a global privacy risk estimation. In order to do so, we calculate the weighted average of the different existing privacy risks multiplied by the maximum risk reduction factors induced from the different blurring components and their settings.

More formally: let Y be the set of all the services y connected to the platform. X_y denotes the set of the sensors connected to a service y, let R_x denotes the set of privacy risks present in the knowledge base related to a sensor x in X_y, B_{xy} the set of blurring components available on the chain between a sensor x and a service y. Let f_{ij} be the risk reduction factor of the blurring component j on a particular privacy risk i. With $f_{ij} \in [0, 1]$; 1 means that the blurring component j in its current settings does not have any effects on the risk i and 0 meaning it counter-effects the risk totally. f_{ij} is calculated using the privacy counter measure knowledge base and it depends on the current blurring configuration. The risk reduction factor f_i on the risk i is defined as being the minimum risk reduction factor on this particular risk i over all the blurring components in the chain. In other words: $f_i = Min(f_{ij}), j \in B_{xy}$. Let w_i be the weight associated to the risk i in the knowledge base. The privacy risk P_{xy} between the sensor x and the data consumer y is calculated as following: $P_{xy} = (\sum_{i \in R_x} w_i * f_i)/|R_x|$. The privacy risk induced by sharing data to service y is calculated by: $P_y = (\sum_{x \in X_y} P_{xy})/|X_y|$. Finally, the global privacy risk of the platform is defined by: $P = \sqrt{(\sum_{y \in Y} (P_y - T_y)^2)}$, where T_y is the tolerated privacy risk for the service y defined by the data owner. P will serve as the fitness function representing the privacy risk to minimize by the evolutionary algorithm. This is presented in the next section.

9.3.3 Reasoning Engine

The Evolutionary Algorithm Reasoning Engine (EARE) is the core component of our platform. As depicted in the framework architecture section (section 9.3), it continuously monitors context changes, service requirements, and user privacy preferences. EARE leverages the reflexion ability of the models@run.time layer and performs adaptations through the model. Whenever a new service connects to our platform, the reasoning engine analyzes the following information: *Technical information* of the service, *e.g.* IP address, port, and user credentials. *Performance requirements* and *communication preferences*. This is inspired from Android's permission based applications [WGNF12]. *User's privacy risk tolerance* for this service or group the service belongs to and for the current context. Different *fitness functions* that represent requirements for privacy, data utility for the service, and performance. In future work we plan to integrate additional fitness functions, *e.g.* to optimize energy consumption or network quotas. Therefore, we added an *extensibility feature* to allow to provide additional fitness function implementations.

Our reasoning engine can access the architecture model representing the current running platform and modify it using models@run.time concepts. In order to setup a new connection for a service, the EARE first deploys a proxy and configures it with the technical configurations of the service in question. Then, a search starts to find the best path for each sensor to this proxy in terms of blurring components. The initial population is seeded with empty paths (aka all required sensors are connected directly to the proxy without blurring in the middle). At each step of the optimization a new generation of potential solutions is generated by mutating the previous generation.

We define three possible mutation operations: *1)* adding a blurring component between a sensor and the proxy, *2)* removing an existing blurring component from the chain, and *3)* tuning the intensity parameter of an existing blurring component. Using just these three mutations, we are able to generate any possible architecture of blurring components. The Java code in listing 6, shows the implementation of the third mutation operator, which randomly selects a sensor, then a blurring component attached to this sensor and mutates the corresponding intensity parameter of the blur between the range [min,max].

After the mutation step, we evaluate the generated solutions of the new population against fitness functions. We provide the following three fitness functions: First, a fitness function that uses the risk and counter-measure knowledge base to evaluate the privacy risks of a particular blurring chain. Second, a fitness function that evaluates the data consumer requirements and the utility of the data sent. And finally, a fitness function that aims to optimize the performance, thus minimize the time lag between sensors and gateways. A blurring component with long processing time will be penalized according to this fitness.

The evolution stops whenever there is a stable state of fitnesses or a maximum number of generations have been reached. At this point, several possible architectures of blurring components between sensors and services are computed. In the final surviving population each representing a different possible trade-off. Our reasoning engine selects the one with

Algorithm 6 Example of Model-based genetic mutator

```
class ChangeBlurSettingMutator implements MutationOperator<KevModel> {
void mutate(KevoreeModel model) {
List<ComponentInstance> ls = getSensors(model);
//Select sensor randomly and get all the blurring components attached
ComponentInstance sensor= ls.get(random.nextInt(ls.size()));
List<ComponentInstance> already= getAttachedBlurComp(model, sensor);
if(already.size()>0){
//Select randomly a blurring comp. and change param
ComponentInstance cmp = already.get(random.nextInt(already.size()));
double min = cmp.getDictionary().find("min").getValue();
double max= cmp.getDictionary().find("max").getValue();
double val = random.nextDouble()*(max-min)+min;
cmp.getDictionary().find("value").setValue(val);
}}}
```

the biggest Hypervolume [ZBT07]. The last step consists of actually deploying the blurring components and the first flow of information is established from our platform to the newly connected service.

In case a new service or user connects to the platform, any user (owner) configuration changes, a knowledge base is updated, or a service preference changes the reasoning engine enters again the optimization search stages. This time, the previously deployed architecture instead of an empty one is used as initial population.

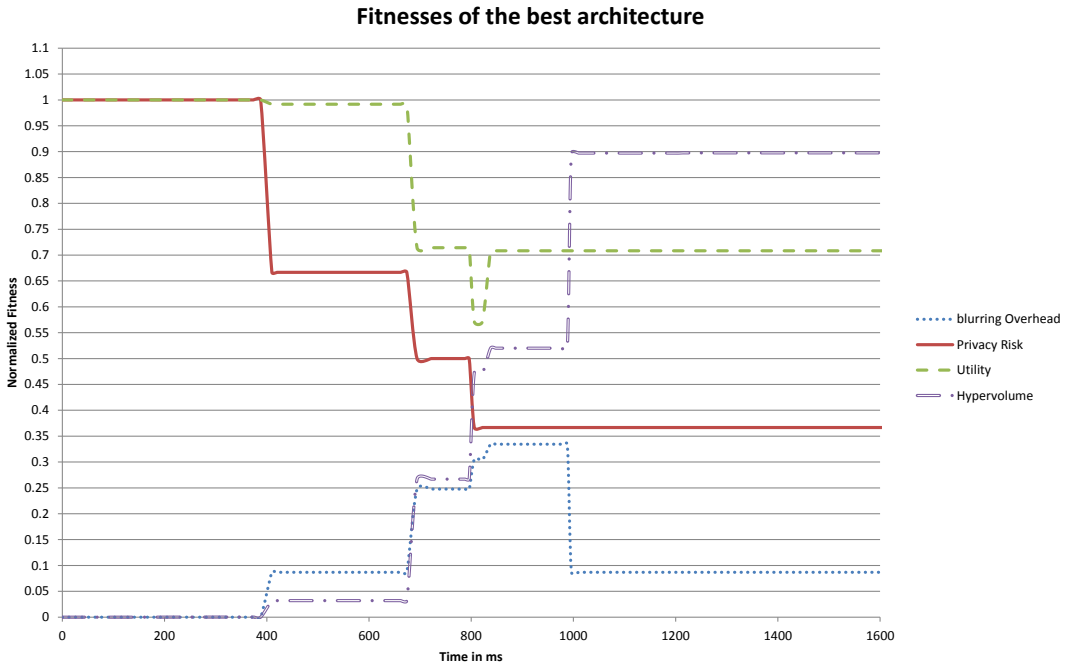

Figure 9.7: Evolution of the different fitness functions and the hyper-volume in time during the search

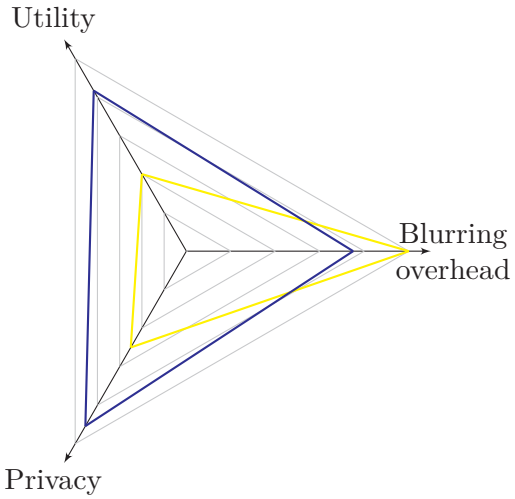

Figure 9.8: Trade-off between fitnesses

9.4 Results and Evaluation

In this section we evaluate our platform on a smart home domain case study. The validation is based on the following key performance indicators: How much time does it require to connect a data consumer for a first time to the platform? How much time is needed for re-adaption in case of context or input changes? How good is the solution found by the evolutionary algorithm compared to a full search?

Setup: Our platform and implementation details are available as open source[ii]. We used 4 sensors from the smart home and health care domain: a temperature sensor, humidity, heart rate, and power consumption. The privacy risks and counter-measures knowledge base were filled from known risks in literature [FVPW07] (12 counter-measures involving different components were setup). We have implemented three fitness functions: the privacy risk fitness, the utility of the shared data fitness (a measure of the information in the blurred stream compared to the original stream [BG06]), the blurring overhead fitness (the time in ms required by the blurring components to perform their tasks). In order to evaluate our approach, we will rely on the following metrics: The first indicator is the time required to connect a new data consumer to our platform for the first time. The second indicator is the time for re-adaptation. It is defined as the time needed from the moment an input is changed (be it the context, the privacy risk knowledge base, the user's preference, or data consumer's settings) to the moment a new architecture is deployed and fully functioning. The third indicator is how well the evolutionary algorithm was able to find good trade-offs compared to a full search based approach. We perform the simulations on an Intel Core i7 2620M CPU with 8Gb of RAM.

Results: To study the performance of the platform, we consider the worst case scenario: the different consumers connected to the platform have asked all for different sensor connection

[ii] https://github.com/securityandtrust/pla

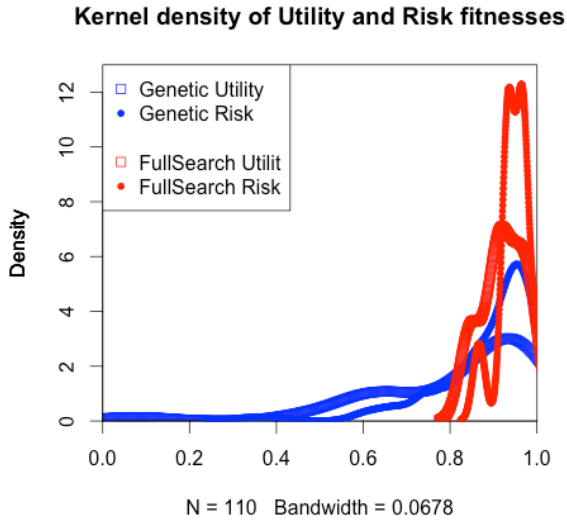

Figure 9.9: Effectiveness of the genetic algorithm compared to full search. On x axis, the fitnesses values (bigger is better). On y, their densities.

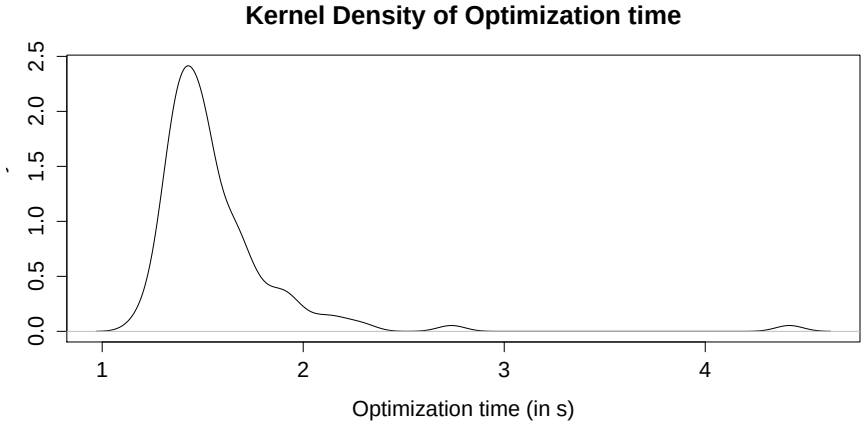

Figure 9.10: Execution duration distribution for our reasonning engine

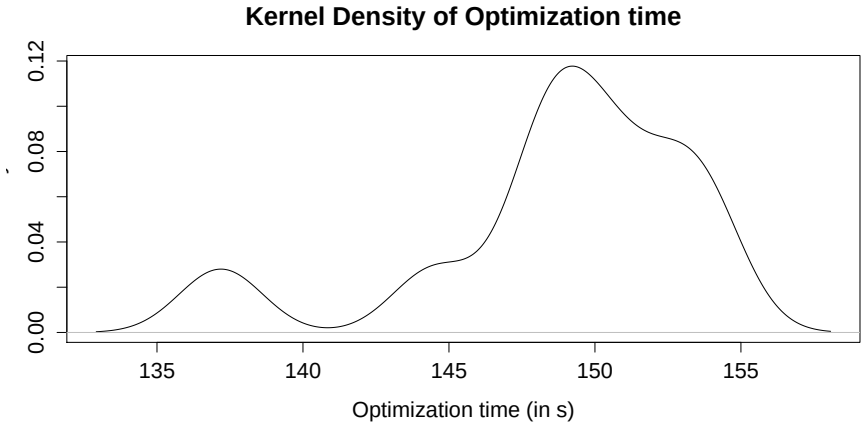

Figure 9.11: Execution duration distribution for a full search

requests and utility preferences. We consider as well that they all have different privacy risk tolerances by the owner. In real-life, consumers may be grouped in privacy groups, thus may have the same privacy settings. Moreover, if they have the same hardware terminals, or want to run the same service, they will require a similar utility level from similar sensors, thus reducing the load on the reasoning engine. After running 100 random tests simulating different user preferences, the average time to connect a data consumer to the platform for the first time was 1.44 seconds (varying between 1 to 2 seconds per data consumer per sensor requested). The density distribution of the optimization time is plotted in figure 9.10. The average time of re-adaptation when an input changes was noticeably smaller (1 second per sensor). The evolutionary search found 80% of the time more than one solution that satisfies the privacy quality within a 5% margin and 90% at least a solution within a 10% margin. By average, 4 architecture solutions were found for each problem. Each architecture represents a different trade-off between the three qualities. The solution with the highest hyper-volume is the one we actually deploy. However, the reasoning engine performed 15% less than a full search in optimizing the utility and privacy risk fitnesses (fig 9.9), but the full search took much more time (150 seconds per consumer per sensor, fig. 9.11). Figure 9.7 shows one run of the evolution in time of the different fitness values of the best architecture model. For this example, we set the tolerated privacy risk level of the new user to 0.4 and the utility requested to 0.7. Initially, the evolutionary search starts with an empty architecture (with no blurring components). Thus, the initial risk fitness is 1, as well as the initial utility of the data (no blurring), and the initial blurring overhead is 0 (no performance delays caused by the non-existence of blurring components). After each generation, the risk fitness function is driven to the requested value of 0.35, which means the addition of new blurring components in the architecture. This reduces the utility of data and increases the blurring overhead. In figure 9.8, we plot the normalized values (between 0 and 1, higher is better, 1 means that the fitness has reached the desired target) of the 3 fitnesses at the initial condition (in yellow) and of the chosen architecture with the best hyper-volume (in blue). We can notice that the blurring overhead fitness has decreased from 1 (no blurring overhead, best performance) but both utility and privacy fitnesses have improved to reached their targets within 10%.

9.5 Discussion

The performance of our platform and its real utility, usefulness, and value depends on the privacy risks and counter-measures knowledge base and the fitness functions provided to the evolutionary algorithm. At one extreme, if this knowledge base is empty, the genetic algorithm will not deploy any blurring between any sensor to any data consumer. On the other hand, the more complex the knowledge base, the more time the evolutionary search engine will need to find a suitable architecture. This can be problematic in case of an emergency situation, to take some time before re-adapting. A simple solution is to pre-calculate and store the architectures to deploy for emergency cases. This dramatically drops the needed time for re-adaptation. Besides, the current privacy risk and counter-measure model treats each sensor independently: each sensor has its own set of possible privacy risks

independently of the other sensors. We did not yet include privacy risks that occur when sharing information with a data consumer from two or more sensors at the same time.

9.6 Conclusion

In this chapter, we presented a concept of proportional access based on blurring techniques in order to dynamically balance privacy and utility for ubiquitous platform services. We showed that this generic concept can be used in ambient assisted living and demonstrated its usefulness for sensor communication platforms. Our platform applies models@run.time as a main paradigm to enable a dynamic reconfiguration whenever the context changes. A new architecture model is derived using an evolutionary multi-objective reasoning engine, which searches the most appropriate solution of blurring components and parameters that best satisfies the qualities regarding privacy, data utility, and efficiency. This allows sensor platform owners to share their data in a controlled manner with data consumers.

Any change in the context will trigger the evolutionary search to dynamically re-adapt the platform and the blurring components (at runtime) to match the new qualities. The privacy risk evaluation in our platform relies on a privacy risk and counter-measure knowledge base. Many previous research results in this domain can be integrated in our knowledge base. For instance, privacy in video surveillance [WAD+05], [NGB03], smart metering [RSMP11], and location data [MCA06].

The evaluation of our platform showed that the overhead introduced by our approach is around 1.5 seconds per user connected and per sensor. The data utility and the blurring overhead were optimized to around 85% of a full search algorithm but is 100 times faster. With the blurring components, and the MOEA algorithms applied to component models, we achieve the adaptability goal of AmI platforms. Each time the context change, an AAL agent can lunch an MOEA optimization search to adapt the blurring chain connected between its sensors and third parties. The last step in this dissertation, is to investigate how to model and include machine learning reasoning for AmI, in order to automatically detect contextual information and automatically profile the qualities to optimize per context.

10

Model-based machine learning

The last contribution of this thesis, aims to address the unobtrusiveness challenge of AmI systems. These systems are supposed to be as transparent as possible for the final end users, meaning that they should be smart enough to automatically detect contexts and habits with the least possible interventions of the users or the system designers. In order to achieve this, we open up perspectives on how to integrate machine learning algorithms directly in the domain specific models. This bridges the gap between the modeling world and the mathematical world of machine learning techniques.

This chapter is a current ongoing work that will continue after the PhD. However, some parts have been already published in the following paper:

- Thomas Hartmann, Assaad Moawad, Francois Fouquet, Yves Reckinger, Tejeddine Mouelhi, Jacques Klein, and Yves Le Traon. Suspicious electric consumption detection based on multi-profiling using live machine learning. In *Smart Grid Communications (SmartGridComm), 2015 IEEE International Conference on*. IEEE, 2015

Contents

10.1 Introduction

So far, we supposed that the context rules and the qualities to optimize per context are manually fed to the AmI system either by system designers or by the user themselves. In order to make AmI systems fade in the background, these parameters need to be learned automatically with the least intervention of the user or system designers. In order to achieve this, the last contribution of this thesis, is to open the roadmap on how to integrate machine learning techniques with models@run.time.

The connection between contextual-reasoning and machine learning is in two directions. On one hand, machine learning can be used to detect context rules, and the qualities to optimize for each context, as shown in figure 10.1. On the other hand, the context information can be used to better train other machine learning algorithms. For instance, profiling user habits or electrical consumptions will be improved when context information is taken into account. For example, knowing if the current day is a holiday or a working day, can affect a lot electrical load prediction.

In this chapter, we present the benefits of model-based context-aware machine learning, as well as a modeling language for machine learning that we implement in KMF. However, we present and evaluate our approach just on the second direction, where we use case from the smart grid domain. We show in this use case, how to use our modelling language in the machine learning domain, and we show the benefits of contextual reasoning for the machine learning.

The first direction, which is the usage of machine learning for context awareness, is currently an ongoing work. It is not mature enough to be included in this chapter, it will be discussed in the future work of this thesis. However, this chapter presents the modelling ideas and the cornerstones on how to prepare an infrastructure and the necessary framework to enable machine learning reasoning for context awareness, thus the title of the thesis is **Towards Ambient Intelligent Applications using Models@run.time and Machine Learning for Context-Aweness**.

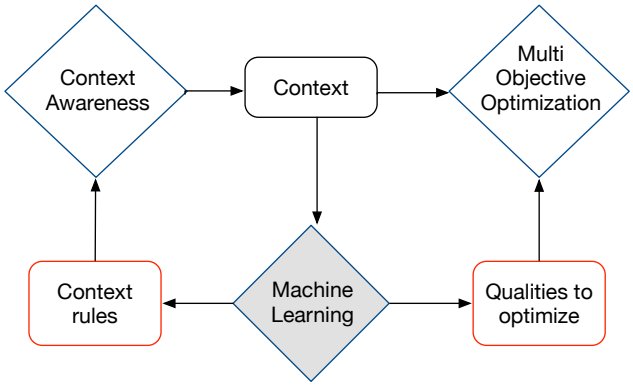

Figure 10.1: Integration of machine learning tools for automatic context detection.

10.2 Model-based machine learning

The central idea of the model-based approach to machine learning is to create a custom model tailored specifically to each new application. In some cases, the model (together with an associated inference algorithm) might correspond to a traditional machine learning technique, while in many cases it will not. Typically, model-based machine learning will be implemented using a model specification language in which the model can be defined using compact code, from which the software implementing that model can be generated automatically [Bis13].

10.2.1 Main principles

The key goals of a model-based approach include the following ideas:

- Segregation of the learning concerns: if the application requires a combination of clustering and classification in the context of time-series data, it is not necessary to mash together traditional algorithms for each of these elements, but instead a single, integrated model capturing the desired behavior can be constructed from several independent building blocks. The modelling language should be able to cascade/chain the different learning models. Therefore, we hope to reuse good structural properties of models such as lazy loading and scalability, which can benefit a lot the machine learning world as well.
- Segregation between the model dependencies and the inference algorithm: the model should specify the domain specific entities that the learning task depends on, independently from the inference algorithm itself. In this way, the inference algorithm can be changed easily without the need to change the interface for the domain application.
- Segregation between model dependencies and attributes extraction. The modeling should allow, for the same dependencies, to extract different machine learning attributes. For example, profiling of electricity consumption depends on smart meter and temperature sensor data. However, many machine learning mathematical hypothesis models can be built using these same dependencies, for example by using the square value of the temperature, or the cube value of the consumption as learning attributes, or trying different combinations of powers of these values. For the domain application, this process should be transparent.
- Re-usability of the learning models: the same learning instance can be used in several tasks without the need to duplicate it several times. For example, in the smart metering domain, an electricity consumption profile of a user can be used in several tasks: to predict the electrical load, to classify the user according to his profile and to detect anomalous consumption behavior. On this point, modeling techniques, such as DSML, offer solutions to define a meta structure that can be reused to define new models of machine learning.

- Transparency of functionality: the model is described by compact code within a generic modelling language, and so the structure of the model is readily apparent. Such modelling code can easily be shared and extended within a community of model builders.

- Combining one learning model from several sub-models: this is usually known as **ensemble methods**. It is used to allow ml models to be composed of multiple weaker ml models that are independently trained and whose results are combined in order to make the overall learning better. There are several way to divide the sub-models. Some techniques splits the training data, this is called *bagging*, other techniques split the features, and other techniques split both data and features (for example random forests).

10.3 Use case: profiling in the smart grid

In order to test the ideas presented in this chapter, especially the modelling of machine learning part and the benefits of context awareness for the ML domain, we use an example from the smart grid domain, based on the smart grid test deployment in Luxembourg. This test deployment contains three different regions, around 300 smart meters, three data concentrators, and a central system. The grid topology in Luxembourg is based on a power line communication [GSW11] (PLC) network and is representative for such smart grid topologies. A more detailed description and analysis of the smart grid topology in Luxembourg can be found in [HFK+14]. The main topology devices in the context of this work are:

- *Smart meters* are the cornerstones of the smart grid infrastructure. Installed at customers' houses they continuously measure electric consumption and quality of power supply and regularly report these values to utilities for monitoring and billing purposes. In Luxembourg, smart meters send the consumption values every 15 minutes for electricity. Another important task of smart meters is load management, as they are able to trigger relays to connect/disconnect specific loads.

- *Data concentrators* collect and store consumption data from a number of associated meters. In regular intervals (several times a day) they send this data, usually via IP connections, to a central control system.

- *Central system* concentrators send their data to a central system where all data are stored, aggregated and analyzed. Because of legal regulations these data must be deleted in regular intervals.

Given the topology structure, an alerting system for suspicious consumption data could be deployed either on the smart meter level (one instance per smart meter), on the data concentrator level (one instance per data concentrator), or at the central system (one global instance).

However, the topology is dynamic and changes over time, for example if a physical cable connection is broken, or in cases of weak signals. In such cases, smart meters dynamically reorganize the topology by reconnecting to other smart meters or concentrators [HFK+14]. Thus the time modeling and learning at runtime is required in this situation, to derive a collective profile at the concentrator level, according to the current topology, and take decisions near real time, before the topology changes again. Moreover, we show that our technique can be used in near real-time and that the contextual information can be used in order to improve the classification.

10.4 Suspicious consumption value detection

In this section we describe our approach for suspicious consumption value detection. First, we present an overview of the approach and the involved components. Next, we detail an explanation of the Gaussian mixture model. Finally, we detail how we learn multiple, context-dependent customer profiles, and how we select and use them for suspicious consumption value detection.

10.4.1 Overview: Towards Contextual Learning and Detection

Figure 10.2 shows a basic overview of our approach. Smart meters continuously report their consumption values, in regular intervals, to utilities for monitoring and billing purposes. In Luxembourg, these intervals are 15 minutes for electricity and 60 minutes for gas. In order to detect suspicious consumption values, this continuous stream of smart meter measurements is first analyzed by a context solver. For every new value, the context solver selects the most appropriate profile for this customer in order to decide what is the normal range of the current measured electricity consumption. For example, for a measurement value on a sunny Monday afternoon (working day) in summer, the context solver will select a profile with comparable context parameters for this customer. The context, for example, can include features like: user type (individual, family, industry, commercial), temporal context (season of the year, month, weekday, holidays), and so forth. The context resolution yields a list of positive and negative profiles. Positive profiles contain information to judge if the measured value is known to be in the normal range, while negative profiles are known to be suspicious values. From these profiles, a decision making step is executed, based on a confidence rate and the probability distribution provided by the profiles. If the measured value is accepted, it is used to train the positive profiles. If it is not accepted, an alert is created. In addition, an interactive validation request is raised to manually validate if the value is indeed suspicious or not. Finally, the negative rated values (suspicious values) are used to train the negative profiles. An initial training period is required to bootstrap the profiles. The update of profiles is a very incremental step and can therefore be performed for every value (as we will show in Section 10.6). However, it would be also feasible to combine several values instead of checking every value.

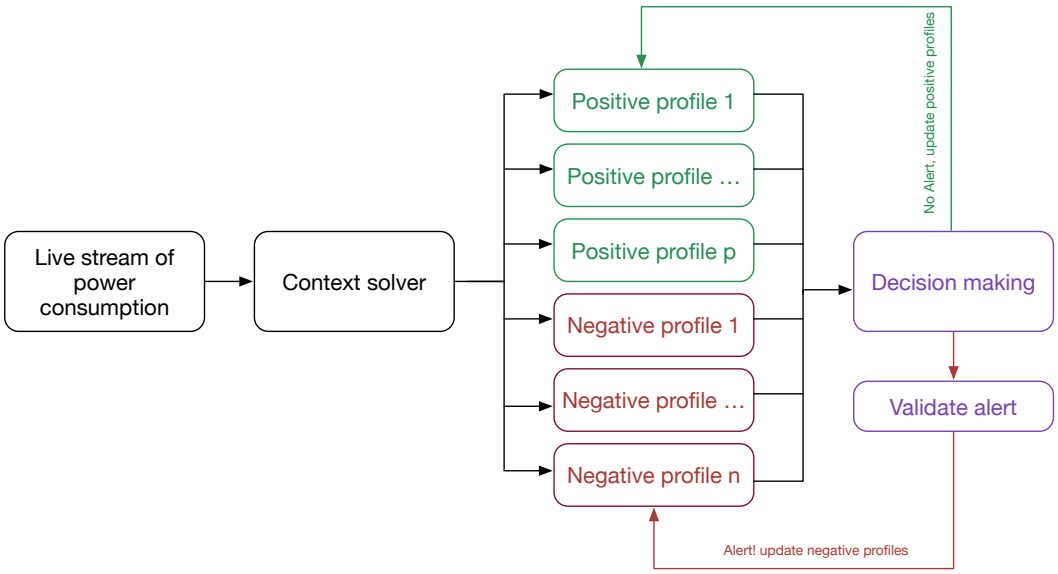

Figure 10.2: Suspicious consumption value detection overview

10.4.2 Gaussian Mixture Model

For this use case, we explore modeling power consumption usage by probability density functions (pdf) based on kernel density estimates (KDE). Particularly, we use Gaussian mixture models (GMM), which are known to be a powerful tool in approximating distributions even when their form is not close to Gaussian [WJ94]. A GMM is a probabilistic model that assumes that all data points are derived from a mixture of Gaussian distributions with unknown parameters. Mixture models are basically generalizing k-means clustering.

Definition 4 *In a nutshell, A Gaussian mixture model of M components, provides the following probability distribution function of an event x happening:*
$p(x) = \sum_{j=1}^{M} w_j K_{\sigma_j}(x)$, *with*
$K_{\sigma_j}(x) = (2\pi\sigma^2)^{-1/2} \exp^{-(x-x_j)^2/(2\sigma^2)}$.

$K_{\sigma_j}(x)$ is one Gaussian component of the mixture model, with an average of x_j, a standard deviation of σ_j and a weight w_j. These parameters must be learned from the data on the fly. Kristan *et al.*,[KL10] provides an online learning algorithm called *oKDE* that is able to update the gaussian mixture model in real-time. The implementation of our software monitoring and alerting system is based on this algorithm, which is suitable and fulfills the requirements we presented in chapter 5 for the live machine learning.

10.4.3 Profiling Power Consumption

In order to build consumption profiles in real-time, we feed the measured consumption values from smart meters to our profiler and process them online. In Luxembourg, each

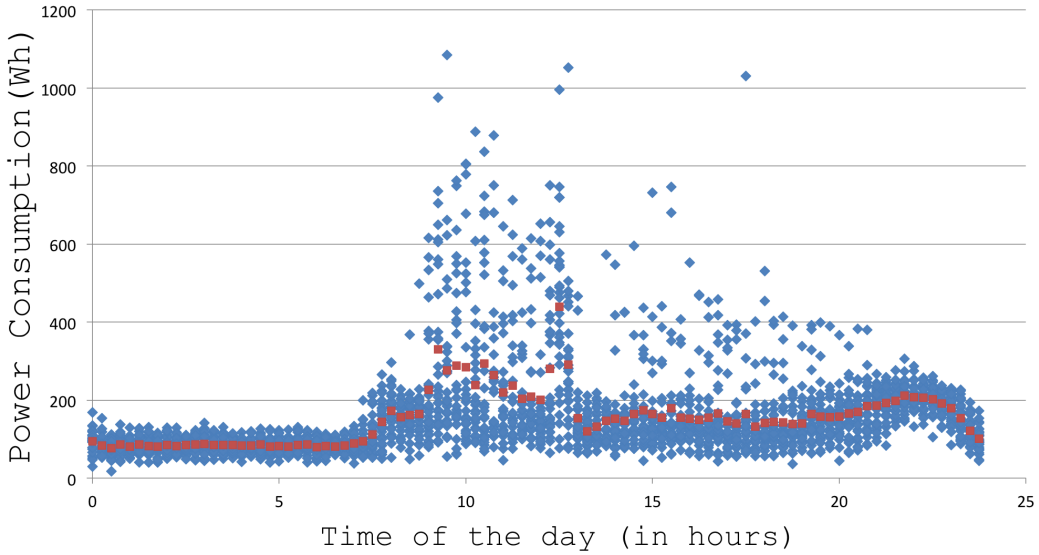

Figure 10.3: Power consumption measures (in blue) and average values (in red)

smart meter reports its consumption values every 15 minutes for electricity (and 60 minutes for gas). Figure 10.3 shows an example of measurements from one customer (one smart meter) over a period of 24 hours. The measurements were taken on 31 weekdays (Monday to Friday). The x-axis in the figure represents the time of the day and the y-axis the consumption (active energy consumed) in *Wh*. Every blue point in the figure corresponds to one measurement value. For example, if we take the time 6.00 am, every point along the y-axis belongs to one measurement for one day at 6.00 am. In red, the figure shows the average for every time, *i.e.,* the average value over all days at every measured time (15 minute intervals). Based on these measurements we can create our consumption profiles for this customer by feeding these measures to our profiler (in real-time) and processing them online. For every new value the consumption profile of the customer will be refined (recalculated). Figure 10.4 shows the profile (the Gaussian mixture model), constructed for the example of figure 10.3.

As an example, the power usage of this user is quite predictable at midnight (varying between 0 and 200 Wh). This is reflected in the profile by a Gaussian Kernel with low variance and we are quite confident (with high probability) that the next midnight measure will be also between 0-200 Wh. However, if we compare this with the consumption at noon, where the user consumes between 0 and 1000 Wh, the profiling has a higher variance, the probability is distributed over a wider range, and thus the prediction is less accurate. In such situations, having a contextual profiling can help to significantly increase the accuracy of the prediction. For instance, during weekends at noon, the consumption may be varying in a less wider range than during weekdays at noon.

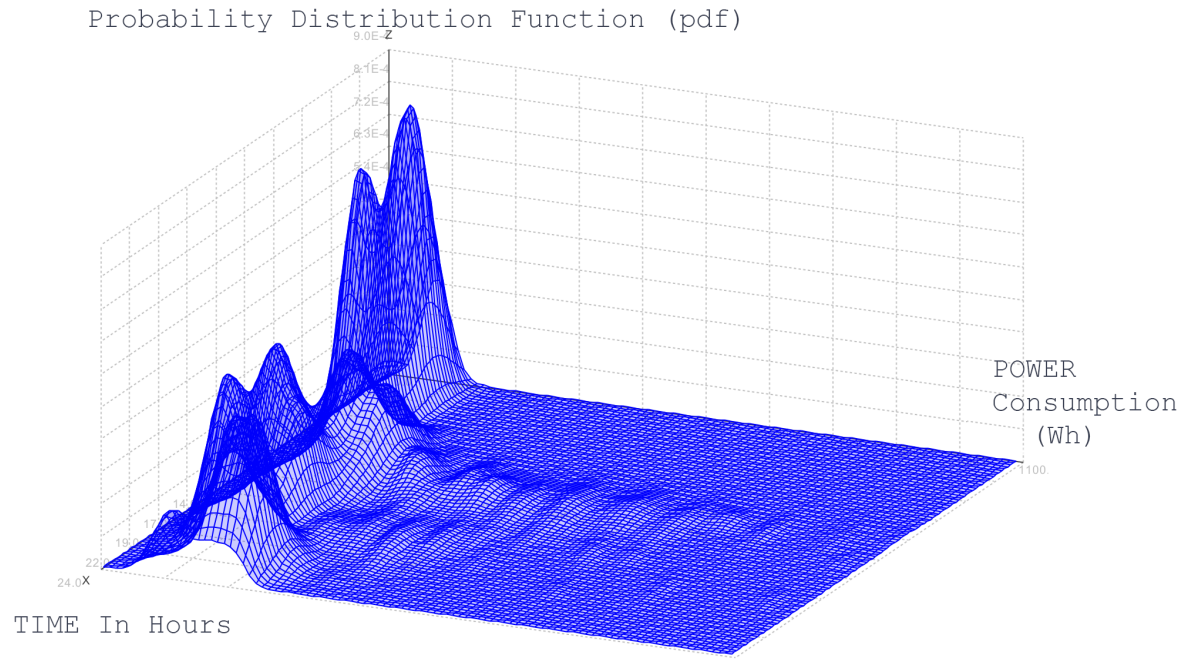

Figure 10.4: Probability distribution function (pdf) of the consumption values from figure 10.3 built with live machine learning

10.5 Modeling language for machine learning

In KMF, we define several keywords to model classes related to machine learning. The syntax of our modeling language includes the following keywords:

- **With inference**: To define the inference or machine learning algorithm to be used in this class.
- **Temporal Resolution**: To define the time span of the learning. After this time interval, the instance of the learning gets cloned automatically and the training values are fed to the new clone. This is done in order to enable meta learning algorithm to inspect the learning itself and check how it is improved over time, and tune the learning parameters accordingly.
- **Dependency**: To define on which other domain specific classes the learning depend on.
- **Input**: To define attribute extractions from the predeclared dependencies, and the mathematical expressions to perform on these attributes as a preprocessing step. Following the main principles we stated before, the segregation of attributes extraction and dependencies declaration is important. For a domain specific application, developers need to provide only the dependencies to the learning class. Moreover, we consider the attribute extraction from these dependencies as meta-learning. We implemented as well a Math expression engine evaluator, to allow any mathematical expression to be declared as a pre-processing step on this input attribute.

- **Output**: To define the expected output from the learning class, in term of domain specific format. KMF manages automatically the dual way conversion between the domain specific concepts and the mathematical representation needed for the learning algorithms.

As a direct concrete modelling example, here is a meta model example from the smartgrid domain.

Algorithm 7 Meta Model of profiling class in smartgrid

```
class smartgrid.SmartMeter{
att maxAllowedPower: Long

att activeEnergyConsumed: Double
att activeEnergyProduced: Double
att reactiveEnergyConsumed: Double
att reactiveEneryProduced: Double

rel profiler: smartgrid.ConsumptionProfiler with maxBound 1
rel classifier: smartgrid.ConsumptionClassification with maxBound 1
}

enum smartgrid.ConsumptionState {
CORRECT, SUSPICIOUS
}

class smartgrid.ConsumptionProfiler {
with inference "GaussianProfiler" with temporalResolution 2592000000
dependency smartmeter: smartgrid.SmartMeter

input timeValue "@smartmeter | =HOURS(TIME)"
input activeEnergyConsumedValue "@smartmeter | =activeEnergyConsumed"
input reactiveEnergyConsumedValue "@smartmeter | =reactiveEnergyConsumed"
//input mixedAttribute "@smartmeter | =reactiveEnergyConsumed * activeEnergyConsumed"
//input cubicAttribute2 "@smartmeter | =reactiveEnergyConsumed^3"

output probability: Double
}

class smartgrid.ConsumptionClassification{
with inference "GaussianClassification" with temporalResolution 2592000000

dependency profiler: smartgrid.ConsumptionProfiler
input probability "@profiler | =probability"

output classification: smartgrid.ConsumptionState
}
```

This meta-model shows an example of use case of the modeling language we are building for machine learning. For every smart meter, we create a relation to a profiler and classifier. The profiler has a dependency to the smart meter, which means that an instance of smart meter is needed to train the profiler, and this instance provides all the attribute extractions. Then, every input line, is one attribute extraction. In a first step, we specify from which dependency the extraction will happens, in the next step we execute a Math expression as a preprocessing of the attribute. In the given example, the function HOUR extracts the hour from the time, and in the comment lines, we perform a multiplication or a cubic power on the attribute. The output line provides the expected attribute from this learning class.

For the classification example, the dependency is for a profiler, and the output will be an

enumeration of type *ConsumptionState* which is either correct or suspicious. Note that, even if we change the algorithm, currently a GaussianClassification, nothing will change for the business domain of electrical consumption. The classification dependency will always be a profiler and the output will always be an enum correct/suspicious whatever machine learning algorithm we use. Moreover, this is an example of how we can separate several machine learning concern in several classes.

In order to enable context aware profiling, we create an additional class of context solver with multiple profiling instances for the same smart meter that dispatches the smart meter value to the profiles resolved according to the context. The same process happens for the classification.

10.5.1 Live meta-learning with meta-modeling@run.time

Meta-learning is about to learn the parameters of the learning class itself and adapt these parameters to the specific business domain where the learning is applied to. The following points can be considered from the meta-learning domain.

- Changing the inference algorithm.
- Adding or removing more input attributes.
- Modifying the math expression (pre-processing) of an attribute.
- Changing learning parameters for example a learning rate alpha.
- Chaining or averaging several learning classes.

These changes can be performed at runtime on the meta-model, using KMF capabilities. For the application domain, the learning interface remains the same. The application domain should only provide instances of the dependencies required for the learning. The learning parameters, algorithms and attribute extractions are internally modeled in the meta-model and can be changed using the reflexive capability of KMF at runtime.

We can imagine as well a scenario where the meta-learning is driven by **an MOEA search at runtime** to optimize the performance of the learning. For instance, several learning meta-models can be generated as an initial population, they can be trained in parallel with live data, and one or several learning metrics performances can be used as fitness functions for the MOEA. The learning meta-classes that have poor performances, will be discarded and replaced by new meta-classes after performing a mutation operator on one or several meta-learning attributes.

These ideas are currently under research, and can open several perspectives in self-adaptation of machine learning classes on the fly. The meta-learning can be performed as well online using the reflexive layer of KMF and the meta-modeling capability at runtime.

10.6 Evaluation

In this section we present the evaluation of our proposed consumption monitoring system. We first describe the setup of our evaluation. Next, we validate the efficiency of our approach, followed by a validation of its performance, *i.e.,* accuracy, precision, recall, and F1 score. We compare these results with a traditional, single profile per user approach.

10.6.1 Experimental Setup

We evaluated our consumption monitoring system based on real data from the Luxembourg smart grid test deployment. The data are provided by our industrial partner Creos Luxembourg S.A. We analyzed consumption data values from a total of 218 smart meters from three different regions in Luxembourg. For our evaluation we considered the consumption data for a time frame of six weeks for each meter. In Luxembourg, each meter reports its consumption data every 15 minutes, via a data concentrator, to the central system. This results in 804345 consumption values in total.

In order to build multiple profiles we considered three different context parameters in our experiments:

- Customer types: since the electricity consumption for residential and industrial customers differs to a significant extent, we clustered customers in two different groups: residential and industrial customers.
- Business day context: we separated working days from weekends and holidays. The idea behind this is that electricity consumption should highly depend on this context parameter. We further divided a day into 96 time slots (=24x4, corresponding to each of the 15 minute reading intervals).
- Weather condition: we collected the historical temperature readings (with an hourly resolution) from meteolux[i], the official meteorological service provider in Luxembourg, and use it as our third context parameter.

We then evaluated our approach in terms of performance to validate if our system is able to be used in a live monitoring system (live detection) and in terms of accuracy. We run the experiments on a 2.6 GHz Intel Core i7 with 8 GB of RAM.

10.6.2 Efficiency: Can We Meet Near Real-Time Expectations?

The core live learning algorithm oKDE to update the profiles of the whole dataset for 218 customers and 803645 consumption values takes 0.22 seconds. This is around 274 nanoseconds per consumption value. It is important to notice, that this step only includes the learning itself, it doesn't contain classification, training, nor decision-making. The whole

[i] http://www.meteolux.lu/

processing including classification, training, selecting the correct profile, and decision-making takes in average 1.37 milliseconds of processing time per consumption value. Considering that the interval of consumption reading in Luxembourg is 15 minutes, we are able to process around 656934 consumption readings during one 15 minutes cycle. Assessing the fact that the computation is conducted in a single thread on a classical computer processor (single core on intel i7 processor), we can consider that our approach is fast enough to be used in a live monitoring system. For example, in the case of Luxembourg with approximately 550.000[ii] inhabitants, a standard laptop is sufficient to monitor the consumption values of the whole country in live. However, as a threat to validity, our performance and therefore the associated near real-time suspicious value detection can be significantly impacted by network access. To overcome this risk, the proposed computation can be split into geographical specialized nodes, *e.g.,* one monitoring system per region (on a data concentrator level). Then, dedicated databases can distribute data based on their usage probability on each node. Another threat to validity is that the profile selection also can considerably impact the performance. We plan to evaluate this in detail in future work.

10.6.3 Effectiveness: Can We Better Detect Suspicious Values?

In order to measure the performance of our multi-context profiling, we compare our approach to a non contextual one, leveraging a single profile per customer approach. We first divide our dataset in two subsets: a training subset of 700.000 and a testing subset of 103.645 consumption values. The latter contains correct, meaning not suspicious consumption values. This is verified by domain experts from Creos.

We then generated two additional sets containing suspicious (false) consumption readings:

- Randomly generated in the interval $[max - 3 * max]$, where max is the maximum electricity consumption value for each customer. We select this interval empirically because it is discriminative, due to the fact that both approaches have 100% accuracy to detect very high deviations. The set evaluates the ability of both approaches (our proposed multi-context profiling and a traditional single profiling per customer approach) to detect suspicious (wrong) consumption values.
- Randomly generated false data between $[max/2 - max]$. This set should test the discriminative power of context-dependent profiling, compared to a single profile per customer. The false data will be seen very real and acceptable if it is not taken within its context.

For our evaluation we use classical metrics, based on:

- True positives (tp): true readings classified as normal.
- True negatives (tn): true readings classified as alert.
- False positives (fp): false readings classified as normal.

[ii] http://www.luxembourg.public.lu/fr/societe/population/

- False negatives (fn): false readings classified as alert.

We then calculate from these values, the accuracy, precision, recall, and F1 score [Pow11] for both approaches.

Profiler	True positives (tp)	False Negatives (fn)	Accuracy %
Single	102675	970	99.06%
Multi-Context	102695	950	99.08%

Table 10.1: Testing set results from Creos

Profiler	True negatives (tn)	False Positives (fp)	Accuracy %
Single	92386	11259	89.13 %
Multi-Context	96458	7187	93.06%

Table 10.2: Testing set results from Randomly generated in the interval [Max - 3x Max]

Profiler	True negatives (tn)	False Positives (fp)	Accuracy %
Single	47297	56348	45.63 %
Multi-Context	86582	17063	83.55%

Table 10.3: Testing set results from Randomly generated in the interval [Max/2-Max]

Attribute	Single Profiler	Multi-context profiler
True positives (tp)	102675	102695
True negatives (tn)	139683	183040
False positives (fp)	67607	24250
False negatives (fn)	970	950
Precision	0.602	0.808
Recall	0.99	0.99
Accuracy	0.779	0.918
F1 score	0.749	0.890

Table 10.4: A global overvew of results

Table 10.1 shows the different results of the 3 test sets. Both profiling techniques performed very well on detecting true positives from Creos (more than 99% as shown in table 10.1). However, the single profiler performed worse when it comes to false positives. For the randomly generated sets, the multi-context profiler was able to detect much more accurately that the values do not correspond to the current user, especially when it comes to values that are in the range of [Max/2-Max]. This validates the fact that taking the context into account, a value that might look fine for a single profile because it falls in the acceptable range, it might look suspicious for the multi-context profiler. The only draw back is that the multi-context profilers need much more time and data to bootstrap. In fact, the more

contexts and profiles we create per user, the more data are required to reach a stable phase from the learning. Finally, table 10.4 summarizes all tests and provide a single metric (F1 score) to compare both approaches overall. Our multi-context profiler was able to score 18% better than a single profiler.

10.7 Conclusion

Modeling machine learning techniques and integrating them with data models and components models, will allow us to create a smart model to respond for the unobtrusiveness challenge for AmI system, and allows fully or semi-automated context awareness. Moreover, the integration of machine learning in modelling techniques has two directional benefits for the contextual reasoning process. In the first direction, these techniques can help automating the discovery of the context rules and settings by mean of clustering or classification or hypothesis basis. In the second way, context awareness can improve the quality of the results of some machine learning techniques. The first way is still under development and will be the subject of my post doctoral works.

Currently, we deployed and tested the second way with the smart grid example. Our proposed system continuously learns context-dependent consumption profiles of customers, *e.g.,* daily, weekly, and monthly profiles, classifies them and selects the most appropriate one according to the context, like date and weather. We showed that, by learning not just one but several profiles per customer and in addition taking context parameters into account, our approach can minimize false alerts (low false positive rate). We evaluated our proposed consumption monitoring system based on real data from the smart grid test deployment in Luxembourg in terms of performance and accuracy. We showed that our suspicious value detection is fast enough to be used in a live monitoring system (live detection) and that it is, in many cases, superior in terms of accuracy to other approaches, which mostly use only one profile per customer and do not take context parameters into account. We also showed that even for randomly generated values, where our approach can hardly profit from its multiple, context-dependent profiles, we are comparable (or even slightly better) than other approaches.

Although this chapter is not complete in term of goals we aimed to achieve, because we did not currently evaluated how to derive contextual rules automatically using machine learning, although the current evaluation was on the opposite direction, on how to use contextual information to improve machine learning, however, this chapter offers the foundations for modeling machine learning and modeling the meta-learning. We discussed as well, how to meta-learn at runtime. Moreover, the same modeling ideas can be applied on both directions. The next steps after my phd will be dedicated to achieve this missing goal.

Part III

Conclusion

11

Conclusions and Future Work

This chapter concludes the dissertation and presents future research directions. We remind first the different challenges we face in IoT, AmI, AAL, we go through the different steps on how we addressed these challenges and the experimentation results we got. Finally we present the next steps to do and the future research axis that will follow this PhD.

Contents

11.1 Summary

In this thesis, we presented first our main application domains: Internet of things, ambient intelligent systems and ambient assisted living. These domains are considered as the next leap of technology when more sensors become cheaper and more deployed everywhere (home, business, public spaces).

The challenges to address in these domains are:

- Big data: In these domains, more existing sensors means more data generated, this might lead to storage and processing problems.
- Continuous aspect of physical measurements: The models representing physical measurements should take into account the continuous nature of time, versus the discretization that happens because of sampling rate.
- Inaccuracies and loss of values: The loss of values can happen because of loss in network communication, the inaccuracy is inherited from the nature of physical sensors.
- Distributed, dynamic and heterogeneous: We have to take into consideration that our agents have different computation power, storage and efficiency and are distributed.
- Context awareness: the first step in reasoning is to detect the current context. In AmI and especially in AAL, system qualities depend on contextual information.
- Adaptability: Once the context is detected, we need to adapt the platform accordingly.
- Unobtrusiveness: the system should infer most of its settings automatically with the least intervention of the user.

In this thesis, we introduced first, a concept of continuous data models that represents sensor physical data more efficiently and in a continuous manner (in time), in order to address the big data, the continuous aspect and the loss of values challenges. We use a series of polynomial functions computed on the fly to model the continuity of sensed data. Our approach is implemented in the open source framework KMF. During an experimental validation we demonstrated the suitability of this approach to build context models handling millions of values from various IoT datasets. The various benchmarks conducted in this work, lead to the conclusion that this representation can enhance, with a very significant factor (from 20 to 1000), primary operations for intelligent systems such as read or write or random access over a context model.

We then enabled a distributed multi-context resolution and awareness, through our R-Core framework, using a distributed component based on models@run.time paradigm. This addresses the context awareness challenge and allows intelligent systems to derive contextual information in a complete distributed fashion using defeasible logic rules. In a first step, we considered contextual rules manually entered by user or system designers. We developed an open source solution on top of Kevoree that allows multi-context resolution

and automatically detects and manages conflicts in term of infinite query loops or incomplete or contradictory contextual information. We tested our query platform till 500 connected agents.

Once the context is derived, an intelligent system should be able to adapt itself accordingly. In order to achieve this, the third contribution of this thesis is to enable multi-objective optimization to run directly on top of domain-specific models without the need of any specific genetic encoding. More specifically, our polymer framework enable MOEAs to run directly on top of component-based models. We presented as well, the blurring components as our privacy preservation elements. The components can act either on the value of the data or on its frequency, to preserve the privacy of the user. However, if the blurring is intense, it might affect the overall performance of the system, or reduce the utility of the sent information. In order to find a good trade-off between these different system qualities, we use Polymer framework as our main reasoning engine. It finds the optimal optimal blurring components to add between an AmI agent and a third party, and tune the settings of these blurring components according to the current context in order to achieve the closest result to the requested trade-off. We show that our framework is responsive for

Finally, we open up perspective on how to integrate machine learning on top of models@run.time in order to derive contextual rules, or in order to automatically profile user habits and or preferred qualities according to the current context. The integration of machine learning in AmI systems allow these system to be unobtrusive, meaning that they will be able to detect and learn usage habits with the least intervention of the user. On the same time, in the other direction, the context awareness aspect in AmI can efficiently improve the machine learning algorithms in certain situations, as we showed in the smart grid domain for suspicious electricity consumption values.

This thesis tackle several interdisciplinary domains, and especially shows the benefit of combining several approaches together to solve real AmI problems. We showed for instance, how MOEA can be used on top of models@run.time to optimize directly the components of an AmI system, in order to find a good trade-off between the several system qualities. Moreover, machine learning algorithms can derive automatically context rules through classification or clustering, and vice versa, contextual reasoning can significantly improve the performance of some machine learning algorithms.

11.2 Next steps and future research axis

Following the same interdisciplinary approach, the next steps of this thesis will be to investigate more in depth the notion of meta-learning using models@run.time. MOEAs can be used to optimize the meta-learning attributes selection at run.time.

A second axis to investigate, is the automatic context discovery rules by monitoring sensor data and user interactions. For this case, pattern detection techniques and periodicity checking mechanisms are needed to detect repeated behavior. This learning has to be done online as well and in near real-time.

A third axis concerning MOEA, is to experiment and improve more the model cloning, so the mutation step is more efficient, especially when only few parts of the model change from one generation to another. For this reason, we are investigating ideas on how to efficiently create a lazy clone from a parent using a notion of parallel universes in KMF. In another direction, we are researching on how to prioritize the different objectives and limit the searching space range per objective according to the context by using a gaussian kernel filter that removes automatically the solutions that are far from the requested results.

A forth research axis, is to investigate how to model and distribute reinforcement learning algorithm for big models, where the search space and the possible combination of states and potential actions are huge. These algorithms can be very useful in order to offer a best policy, aka. a list of actions to take, once the AmI is in the current context or state, in order to maximize some long term goal. This will enable prescriptive analysis in the AmI domain where the system not only learn some user habits, but might take some initiative in suggesting actions to do.

On the more applied side, with the setup of the ongoing setup IoT lab at the university of Luxembourg, we will be able to fully test an overall integrated system. In this thesis, we tested every part on its own, but not a fully developed application from A to Z. There are already several sensors already recording data in the lab for this purpose, and the contributions of this thesis will be applied in a full demo, once we have all the hardware required, and a good historical dataset of values to work with.

As another development task, we started creating common interfaces in KMF for math libraries in order to take advantage of the heterogeneous nature of AmI systems. As the machine learning rely heavily on linear algebra operations, we are currently creating a common interface for BLAS (Basic Linear Algebra Subprograms) and LAPACK (Linear Algebra PACKage). These two famous math libraries, will allow us to take advantages of the presence of a graphic processing unit (gpu) on the host device. The machine learning and math part will be executed more efficiently, faster and in a parallel way.

List of papers and tools

- 2015:
  - Efthymiou Vasilis Bikakis Antonis Caire Patrice, Moawad Assaad and Le Traon Yves. Privacy challenges in ambient intelligence systems. *Journal of Ambient Intelligence and Smart Environments*, 2016
  - Thomas Hartmann, Assaad Moawad, Francois Fouquet, Yves Reckinger, Tejeddine Mouelhi, Jacques Klein, and Yves Le Traon. Suspicious electric consumption detection based on multi-profiling using live machine learning. In *Smart Grid Communications (SmartGridComm), 2015 IEEE International Conference on*. IEEE, 2015
  - Assaad Moawad, Thomas Hartmann, François Fouquet, Jacques Klein, and Yves Le Traon. Adaptive blurring of sensor data to balance privacy and utility for ubiquitous services. In *SAC 2015-The 30th ACM/SIGAPP Symposium On Applied Computing*, pages 2271–2278. ACM, 2015
  - Assaad Moawad, Thomas Hartmann, François Fouquet, Grégory Nain, Jacques Klein, and Johann Bourcier. Polymer: A model-driven approach for simpler, safer, and evolutive multi-objective optimization development. In *MODELSWARD 2015-3rd International Conference on Model-Driven Engineering and Software Development*, pages 286–293. SCITEPRESS, 2015
  - Assaad Moawad, Thomas Hartmann, François Fouquet, Grégory Nain, Jacques Klein, and Yves Le Traon. Beyond discrete modeling: A continuous and efficient model for iot. In *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 90–99. Conference Publishing Consulting, 2015
  - Thomas Hartmann, Assaad Moawad, François Fouquet, Grégory Nain, Jacques Klein, and Yves Le Traon. Stream my models: Reactive peer-to-peer distributed models@run.time. In *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 80–89. Conference Publishing Consulting, 2015
- 2014:
  - Donia El Kateb, Nicola Zannone, Assaad Moawad, Patrice Caire, Grégory Nain, Tejeddine Mouelhi, and Yves Le Traon. Conviviality-driven access control policy. *Requirements Engineering*, pages 1–20, 2014
- 2013:
  - Assaad Moawad, Antonis Bikakis, Patrice Caire, Grégory Nain, and Yves Le Traon. R-core: A rule-based contextual reasoning platform for ami. In *RuleML@ ChallengeEnriched 2013*, 2013
  - Assaad Moawad, Antonis Bikakis, Patrice Caire, Grégory Nain, and Yves Le Traon. A rule-based contextual reasoning platform for ambient intelligence environments. In *Theory, Practice, and Applications of Rules on the Web*, pages 158–172. Springer, 2013

- 2012:
  - Assaad Moawad, Vasileios Efthymiou, Patrice Caire, Grégory Nain, and Yves Le Traon. Introducing conviviality as a new paradigm for interactions among it objects. In *Proceedings of the Workshop on AI Problems and Approaches for Intelligent Environments*, volume 907, pages 3–8. CEUR-WS. org, 2012

Open source contribution during PhD: Kevoree Modeling Framework (KMF): `http://www.kevoree.org/kmf/`

Bibliography

[23793] Ieee standard for a software quality metrics methodology. *IEEE Std 1061-1992*, pages 0,1, 1993.

[ABBZ12] Anne Auger, Johannes Bader, Dimo Brockhoff, and Eckart Zitzler. Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications. *Theoretical Computer Science*, 425:75–103, 2012.

[Abe10] Peter Abeles. Efficient java matrix library. 2010.

[ABGM01] Grigoris Antoniou, David Billington, Guido Governatori, and Michael J. Maher. Representation results for defeasible logic. *ACM Transactions on Computational Logic*, 2(2):255–287, 2001.

[ACRV13] Giovanni Acampora, Diane J Cook, Parisa Rashidi, and Athanasios V Vasilakos. A survey on ambient intelligence in healthcare. *Proceedings of the IEEE*, 101(12):2470–2494, 2013.

[ADB+99] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*, pages 304–307. Springer, 1999.

[AIM10] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

[Alt77] Irwin Altman. Privacy Regulation: Culturally Universal or Culturally Specific? *Journal of Social Issues*, 33:66–84, 1977.

[BA05a] R.J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 217–228, 2005.

[BA05b] Roberto J Bayardo and Rakesh Agrawal. Data privacy through optimal k-anonymization. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 217–228. IEEE, 2005.

[BA10] Antonis Bikakis and Grigoris Antoniou. Defeasible contextual reasoning with arguments in ambient intelligence. *IEEE Trans. Knowl. Data Eng.*, 22(11):1492–1506, 2010.

[BA11] Antonis Bikakis and Grigoris Antoniou. Partial preferences and ambiguity resolution in contextual defeasible logic. In *LPNMR*, pages 193–198, 2011.

[BAH11] Antonis Bikakis, Grigoris Antoniou, and Panayiotis Hassapis. Strategies for contextual reasoning with conflicts in Ambient Intelligence. *Knowledge and Information Systems*, 27(1):45–84, 2011.

[BBF09] Gordon S. Blair, Nelly Bencomo, and Robert B. France. Models@ run.time. *IEEE Computer*, 42(10):22–27, 2009.

[BCDS11] Jeremiah Blocki, Nicolas Christin, Anupam Datta, and Arunesh Sinha. Regret minimizing audits: A learning-theoretic basis for privacy protection. In *Proceedings of the 2011 IEEE 24th Computer Security Foundations Symposium*, CSF '11, pages 312–327, Washington, DC, USA, 2011. IEEE Computer Society.

[BDDO04] Jürgen Branke, Kalyanmoy Deb, Henning Dierolf, and Matthias Osswald. Finding knees in multi-objective optimization. In *Parallel Problem Solving from Nature-PPSN VIII*, pages 722–731. Springer, 2004.

[BDR07] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277, 2007.

[BFCA14] Nelly Bencomo, Robert B. France, Betty H. C. Cheng, and Uwe Aßmann, editors. *Models@run.time - Foundations, Applications, and Roadmaps [Dagstuhl Seminar 11481, November 27 - December 2, 2011]*, volume 8378 of *Lecture Notes in Computer Science*. Springer, 2014.

[BG06] Lakshminath Bhuvanagiri and Sumit Ganguly. Estimating entropy over data streams. In *Algorithms–ESA 2006*, pages 148–159. Springer, 2006.

[Bik09] Antonis Bikakis. *Defeasible Contextual Reasoning in Ambient Intelligence*. PhD thesis, University of Crete, 2009.

[Bis13] Christopher M Bishop. Model-based machine learning. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 371(1984):20120222, 2013.

[BN98] P. Brierley-Newell. A cross-cultural comparison of privacy definitions and functions: A systems approach. *Journal of Environmental Psychology*, 18:357–371, 1998.

[BS03] Alastair R. Beresford and Frank Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, January 2003.

[BTE+05] LISA BROWN, YING-LI TIAN, AHMET EKIN, CHIAO FE SHU, and MAX LU. Enabling video privacy through computer vision. 2005.

[Bur82] J.K. Burgoon. Privacy and communication. *Communication Yearbook*, 6:206–249, 1982.

[CAJ09] Diane J Cook, Juan C Augusto, and Vikramaditya R Jakkula. Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4):277–298, 2009.

[CB03] Mary J. Culnan and Robert J. Bies. Consumer privacy: Balancing economic and justice considerations. *Journal of Social Issues*, 59(2):323–342, 2003.

[CF99] Kin-Pong Chan and AW-C Fu. Efficient time series matching by wavelets. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 126–133. IEEE, 1999.

[CFJ03] Harry Chen, Tim Finin, and Anupam Joshi. An ontology for context-aware pervasive computing environments. *The Knowledge Engineering Review*, 18(03):197–207, 2003.

[Chr12] Peter Christen. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Data-centric systems and applications. Springer, 2012.

[CKD+04] Chris Clifton, Murat Kantarciog(lu, AnHai Doan, Gunther Schadow, Jaideep Vaidya, Ahmed Elmagarmid, and Dan Suciu. Privacy-preserving data integration and sharing. In *Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 19–26. ACM, 2004.

[CNYM12] Lawrence Chung, Brian A Nixon, Eric Yu, and John Mylopoulos. *Non-functional requirements in software engineering*, volume 5. Springer Science & Business Media, 2012.

[Coo78] T.M.I. Cooley. *A Treatise on the Law of Torts: Or the Wrongs which Arise Independent of Contract*. Callaghan, 1878.

[CPY16] Efthymiou Vasilis Bikakis Antonis Caire Patrice, Moawad Assaad and Le Traon Yves. Privacy challenges in ambient intelligence systems. *Journal of Ambient Intelligence and Smart Environments*, 2016.

[CVVL02] Carlos A Coello Coello, David A Van Veldhuizen, and Gary B Lamont. *Evolutionary algorithms for solving multi-objective problems*, volume 242. Springer, 2002.

[Dal77] T. Dalenius. Towards a methodology for statistical disclosure control. *Statistik Tidskrift*, 15(429-444):2–1, 1977.

[Dav97] Simon G Davies. Re-engineering the right to privacy: how privacy has been transformed from a right to a commodity. In *Technology and privacy*, pages 143–165. MIT Press, 1997.

[DD13] Haluk Demirkan and Dursun Delen. Leveraging the capabilities of service-oriented decision support systems: Putting analytics and big data in cloud. *Decision Support Systems*, 55(1):412–421, 2013.

[DeC97] J.W. DeCew. *In Pursuit of Privacy: Law, Ethics, and the Rise of Technology*. G - Reference, Information and Interdisciplinary Subjects Series. Cornell University Press, 1997.

[DFB+12] Erwan Daubert, François Fouquet, Olivier Barais, Grégory Nain, Gerson Sunye, Jean-Marc Jézéquel, J-L Pazat, and Brice Morin. A models@ runtime framework for designing and managing service-based applications. In *Software Services and Systems Research-Results and Challenges (S-Cube), 2012 Workshop on European*, pages 10–11. IEEE, 2012.

[DG03] Curt J Dommeyer and Barbara L Gross. What consumers know and what they do: An investigation of consumer knowledge, awareness, and use of privacy protection strategies. *Journal of Interactive Marketing*, 17(2):34–51, 2003.

[DKM+] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology-EUROCRYPT 2006*, pages 486–503.

[DLM+98] Gautam Das, King-Ip Lin, Heikki Mannila, Gopal Renganathan, and Padhraic Smyth. Rule discovery from time series. In *KDD*, volume 98, pages 16–22, 1998.

[DMOD+10] Angelika Dohr, R Modre-Opsrian, Mario Drobics, Dieter Hayn, and Günter Schreier. The internet of things for ambient assisted living. In *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, pages 804–809. Ieee, 2010.

[DN11] Juan J Durillo and Antonio J Nebro. jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760–771, 2011.

[DPAM02] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE*, 2002.

[Dwo06] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006.

[Dwo08] Cynthia Dwork. Differential privacy: a survey of results. In *Proceedings of the 5th international conference on Theory and applications of models of computation*, TAMC'08, pages 1–19, Berlin, Heidelberg, 2008. Springer-Verlag.

[Dwo11] Cynthia Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54(1):86–95, January 2011.

[EKZM+14] Donia El Kateb, Nicola Zannone, Assaad Moawad, Patrice Caire, Grégory Nain, Tejeddine Mouelhi, and Yves Le Traon. Conviviality-driven access control policy. *Requirements Engineering*, pages 1–20, 2014.

[FBP+12] François Fouquet, Olivier Barais, Noël Plouzeau, Jean-Marc Jézéquel, Brice Morin, and Franck Fleurey. A Dynamic Component Model for Cyber Physical Systems. In *15th International Symposium on Component Based Software Engineering*, Italy, July 2012.

[FDP+12] François Fouquet, Erwan Daubert, Noel Plouzeau, Olivier Barais, Johann Bourcier, and Arnaud Blouin. Kevoree: une approche model@ runtime pour les systèmes ubiquitaires. In *UbiMob2012*, 2012.

[FFH] S. Frey, F. Fittkau, and W. Hasselbring. Search-based genetic optimization for deployment and reconfiguration of software in the cloud. In *Software Engineering (ICSE), 2013 35th International Conference on*.

[Fie06] Stephen E Fienberg. Privacy and confidentiality in an e-commerce world: Data mining, data warehousing, matching and disclosure limitation. *Statistical Science*, 21(2):143–154, 2006.

[FNM+12a] François Fouquet, Grégory Nain, Brice Morin, Erwan Daubert, Olivier Barais, Noël Plouzeau, and Jean-Marc Jézéquel. An Eclipse Modelling Framework Alternative to Meet the models@run.time Requirements. In *Models 2012*, Innsbruck, Autriche, October 2012.

[FNM+12b] François Fouquet, Grégory Nain, Brice Morin, Erwan Daubert, Olivier Barais, Noël Plouzeau, and Jean-Marc Jézéquel. *An eclipse modelling framework alternative to meet the models@run.time requirements*. Springer, 2012.

[FNM+14] François Fouquet, Grégory Nain, Brice Morin, Erwan Daubert, Olivier Barais, Noël Plouzeau, and Jean-Marc Jézéquel. Kevoree modeling framework (kmf): Efficient modeling techniques for runtime use. *CoRR*, abs/1405.6817, 2014.

[FVPW07] Michael Friedewald, Elena Vildjiounaite, Yves Punie, and David Wright. Privacy, identity and security in ambient intelligence: A scenario analysis. *Telematics and Informatics*, 24(1):15–29, 2007.

[FWCY10] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42(4):14:1–14:53, June 2010.

[Gar14] Stephen Prentice Gartner. The five smart technologies to watch, 2014.

[GBMP13] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.

[GBT94] Simon Gibbs, Christian Breiteneder, and Dennis Tsichritzis. *Data modeling of time-based media*, volume 23. ACM, 1994.

[Geo08] Peter Georgieff. Ambient assisted living. *Marktpotenziale IT-unterstützter Pflege für ein selbstbestimmtes Altern, FAZIT Forschungsbericht*, 17:9–10, 2008.

[GG01] Chiara Ghidini and Fausto Giunchiglia. Local Models Semantics, or contextual reasoning=locality+compatibility. *Artificial Intelligence*, 127(2):221–259, 2001.

[Gor92] Ken Gormley. One hundred years of privacy. *Wis. L. Rev.*, page 1335, 1992.

[GPR06] Mykola Galushka, Dave Patterson, and Niall Rooney. Temporal data mining for smart homes. In *Designing Smart Homes*, pages 85–108. Springer, 2006.

[Gri04] Stefanos Gritzalis. Enhancing web privacy and anonymity in the digital era. *Inf. Manag. Comput. Security*, 12(3):255–287, 2004.

[GS94] Fausto Giunchiglia and Luciano Serafini. Multilanguage hierarchical logics, or: how we can do without modal logics. *Artificial Intelligence*, 65(1), 1994.

[GSW11] S. Galli, A. Scaglione, and Zhifang Wang. For the grid and through the grid: The role of power line communications in the smart grid. *Proc. of the IEEE*, 99(6):998–1027, June 2011.

[GvdHT09] John C Georgas, André van der Hoek, and Richard N Taylor. Using architectural models to manage and visualize runtime adaptation. *Computer*, (10):52–60, 2009.

[HAG+13] Martin Hirzel, Henrique Andrade, Bugra Gedik, Gabriela Jacques-Silva, Rohit Khandekar, Vibhore Kumar, Mark Mendell, Howard Nasgaard, Scott Schneider, Robert Soulé, et al. Ibm streams processing language: Analyzing big data in motion. *IBM Journal of Research and Development*, 57(3/4):7–1, 2013.

[HFK+14] Thomas Hartmann, Francois Fouquet, Jacques Klein, Yves Le Traon, Alexander Pelov, Laurent Toutain, and Tanguy Ropitault. Generating realistic smart grid communication topologies based on real-data. In *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, pages 428–433. IEEE, 2014.

[HFN+14a] Thomas Hartmann, François Fouquet, Grégory Nain, Brice Morin, Jacques Klein, Olivier Barais, and Yves Le Traon. A native versioning concept to support historized models@run.time. In *17th International Conference on Model Driven Engineering Languages and Systems (MODELS 2014)*, pages 252–268. Springer, 2014.

[HFN+14b] Thomas Hartmann, François Fouquet, Grégory Nain, Brice Morin, Jacques Klein, and Yves Le Traon. Reasoning at runtime using time-distorted contexts: A models@run.time based approach. In *26th International Conference on Software Engineering and Knowledge Engineering*, pages 586–591. Knowledge Systems Institute Graduate School, USA, 2014.

[HGH+08] Baik Hoh, Marco Gruteser, Ryan Herring, Jeff Ban, Daniel Work, Juan-Carlos Herrera, Alexandre M Bayen, Murali Annavaram, and Quinn Jacobson. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 15–28. ACM, 2008.

[HI04] Karen Henricksen and Jadwiga Indulska. Modelling and Using Imperfect Context Information. In *Proceedings of PERCOMW '04*, pages 33–37, Washington, DC, USA, 2004. IEEE Computer Society.

[HM06] Mireille Hildebrandt and Martin Meints. Rfid, profiling, and ami. Technical report, FIDIS (Future of Identity in the Information Society), Deliverable D7.7, http://www.fidis.net, 2006.

[HM10] Mark Harman and Phil McMinn. A theoretical and empirical study of search-based testing: Local, global, and hybrid search. *IEEE Trans. Softw. Eng.*, 36(2):226–247, March 2010.

[HMF+15a] Thomas Hartmann, Assaad Moawad, François Fouquet, Grégory Nain, Jacques Klein, and Yves Le Traon. Stream my models: Reactive peer-to-peer distributed models@run.time. In *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 80–89. Conference Publishing Consulting, 2015.

[HMF+15b] Thomas Hartmann, Assaad Moawad, Francois Fouquet, Yves Reckinger, Tejeddine Mouelhi, Jacques Klein, and Yves Le Traon. Suspicious electric consumption detection based on multi-profiling using live machine learning. In *Smart Grid Communications (SmartGridComm), 2015 IEEE International Conference on*. IEEE, 2015.

[Hog92] Mark L Hogarth. Does general relativity allow an observer to view an eternity in a finite time? *Foundations of physics letters*, 5(2):173–181, 1992.

[Hon04] Jason I. Hong. An architecture for privacy-sensitive ubiquitous computing. In *In MobiSYS '04: Proceedings of the 2nd international conference on mobile systems, applications, and services*, pages 177–189. ACM Press, 2004.

[Inf14] InfluxDB, 2014.

[Jac09] Adam Jacobs. The pathologies of big data. *Communications of the ACM*, 52(8):36–44, 2009.

[Jue06] Ari Juels. Rfid security and privacy: A research survey. *Journal of Selected Areas in Communication (J-SAC)*, 24(2):381–395, 2006.

[KCHP01] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. An online algorithm for segmenting time series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 289–296. IEEE, 2001.

[KCPM01] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286, 2001.

[KCS06] Abdullah Konak, David W Coit, and Alice E Smith. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety*, 2006.

[KE+95] James Kennedy, Russell Eberhart, et al. Particle swarm optimization. 4(2):1942–1948, 1995.

[Kel11] Daniel A Kelly. Disaggregating smart meter readings using device signatures. 2011.

[KFB11] Georgios Kalogridis, Zhong Fan, and Sagar Basutkar. Affordable privacy for home smart meters. In *Parallel and Distributed Processing with Applications Workshops (ISPAW), 2011 Ninth IEEE International Symposium on*, pages 77–84. IEEE, 2011.

[KG06] Daniel Kifer and Johannes Gehrke. Injecting utility into anonymized datasets. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 217–228. ACM, 2006.

[KL10] Matej Kristan and Aleš Leonardis. Multivariate online kernel density estimation. In *Computer Vision Winter Workshop*, pages 77–86, 2010.

[KL14] Matej Kristan and Ales Leonardis. Online discriminative kernel density estimator with gaussian kernels. *Cybernetics, IEEE Transactions on*, 44(3):355–365, 2014.

[KS02] Günter Karjoth and Matthias Schunter. A privacy policy model for enterprises. In *Proceedings of the 15th IEEE workshop on Computer Security Foundations*, CSFW '02, pages 271–, Washington, DC, USA, 2002. IEEE Computer Society.

[KSL10] Matej Kristan, Danijel Skočaj, and Ales Leonardis. Online kernel density estimation for interactive learning. *Image and Vision Computing*, 28(7):1106–1116, 2010.

[Kup87] Joseph Kupfer. Privacy, autonomy, and self-concept. *American Philosophical Quarterly*, 24(1):pp. 81–89, 1987.

[Lan01] Marc Langheinrich. Privacy by design - principles of privacy-aware ubiquitous systems. In Gregory Abowd, Barry Brumitt, and Steven Shafer, editors, *Ubicomp 2001: Ubiquitous Computing*, volume 2201 of *Lecture Notes in Computer Science*, pages 273–291. Springer Berlin / Heidelberg, 2001.

[LKVD+01] H. Leino-Kilpi, M. Välimäki, T. Dassen, M. Gasull, C. Lemonidou, A. Scott, and M. Arndt. Privacy: a review of the literature. *International Journal of Nursing Studies*, 38(6):663–671, 2001.

[LL09] Tiancheng Li and Ninghui Li. On the tradeoff between privacy and utility in data publishing. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 517–526. ACM, 2009.

[LLMS13] Chao Li, Daniel Yang Li, Gerome Miklau, and Dan Suciu. A theory of pricing private data. In *Proceedings of the 16th International Conference on Database Theory*, ICDT '13, pages 33–44, New York, NY, USA, 2013. ACM.

[LLS+13] Steve LaValle, Eric Lesser, Rebecca Shockley, Michael S Hopkins, and Nina Kruschwitz. Big data, analytics and the path from insights to value. *MIT Sloan Management Review*, 21, 2013.

[LLV07] Ninghui Li, Tiancheng Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 106 –115, april 2007.

[LM09] Xiao-Bai Li and Luvai Motiwalla. Protecting patient privacy with data masking. In *Proceedings of the Fourth Annual AIS SIGSEC Workshop on Information Security and Privacy (WISP 2009)*, Phoenix, AZ, USA, 2009.

[MA04] R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

[Mad12] Mary Madden. Privacy management on social media sites. *Pew Internet Report*, pages 1–20, 2012.

[MBC+13a] Assaad Moawad, Antonis Bikakis, Patrice Caire, Grégory Nain, and Yves Le Traon. R-core: A rule-based contextual reasoning platform for ami. In *RuleML@ ChallengeEnriched 2013*, 2013.

[MBC+13b] Assaad Moawad, Antonis Bikakis, Patrice Caire, Grégory Nain, and Yves Le Traon. A rule-based contextual reasoning platform for ambient intelligence environments. In *Theory, Practice, and Applications of Rules on the Web*, pages 158–172. Springer, 2013.

[MBNJ09] Brice Morin, Olivier Barais, Gregory Nain, and Jean-Marc Jezequel. Taming dynamically adaptive systems using models and aspects. In *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pages 122–132, Washington, DC, USA, 2009.

[MCA06] Mohamed F Mokbel, Chi-Yin Chow, and Walid G Aref. The new casper: query processing for location services without compromising privacy. In *Proceedings of the 32nd international conference on Very large data bases*, pages 763–774. VLDB Endowment, 2006.

[MEC+12a] Assaad Moawad, Vasileios Efthymiou, Patrice Caire, Grégory Nain, and Yves Le Traon. Introducing conviviality as a new paradigm for interactions among IT objects. In *Proceedings of the Workshop on AI Problems and Approaches for Intelligent Environments*, volume 907, pages 3–8. CEUR-WS.org, 2012.

[MEC+12b] Assaad Moawad, Vasileios Efthymiou, Patrice Caire, Grégory Nain, and Yves Le Traon. Introducing conviviality as a new paradigm for interactions among it objects. In *Proceedings of the Workshop on AI Problems and Approaches for Intelligent Environments*, volume 907, pages 3–8. CEUR-WS. org, 2012.

[MHF+15a] Assaad Moawad, Thomas Hartmann, François Fouquet, Jacques Klein, and Yves Le Traon. Adaptive blurring of sensor data to balance privacy and utility for ubiquitous services. In *SAC 2015-The 30th ACM/SIGAPP Symposium On Applied Computing*, pages 2271–2278. ACM, 2015.

[MHF+15b] Assaad Moawad, Thomas Hartmann, François Fouquet, Grégory Nain, Jacques Klein, and Johann Bourcier. Polymer: A model-driven approach for simpler, safer, and evolutive multi-objective optimization development. In *MODELSWARD 2015-3rd International Conference on Model-Driven Engineering and Software Development*, pages 286–293. SCITEPRESS, 2015.

[MHF+15c] Assaad Moawad, Thomas Hartmann, François Fouquet, Grégory Nain, Jacques Klein, and Yves Le Traon. Beyond discrete modeling: A continuous and efficient model for iot. In *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 90–99. Conference Publishing Consulting, 2015.

[MKGV07] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1), March 2007.

[MM09] Patrick McDaniel and Stephen McLaughlin. Security and privacy challenges in the smart grid. *IEEE Security and Privacy*, 7(3):75–77, May 2009.

[Moo03] Adam D. Moore. Privacy: Its meaning and value. *American Philosophical Quarterly*, 40(3):pp. 215–227, 2003.

[NGB03] Carman Neustaedter, Saul Greenberg, and Michael Boyle. Balancing privacy and awareness for telecommuters using blur filtration. Technical report, Report 2003-719-22, Department of Computer Science, University of Calgary, 2003.

[OG+05] Yael Onn, Michael Geva, et al. Privacy in the digital environment. *Haifa Center of Law & Technology*, pages 1–12, 2005.

[Ohm09] Paul Ohm. Broken Promises of Privacy: Responding to the Surprising Failure of Anonymization. *Social Science Research Network Working Paper Series*, August 2009.

[ore13] oreilly. The re-emergence of time-series, 2013.

[Pap10] Zizi Papacharissi. Privacy as a luxury commodity. *First Monday*, 15(8):2–5, 2010.

[PD03] Leysia Palen and Paul Dourish. Unpacking "privacy" for a networked world. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 129–136, New York, NY, USA, 2003. ACM.

[Pla] PlanetCassandra. Getting started with time series data modeling.

[Pow11] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[PPZ11] Milan Petkovic, Davide Prandi, and Nicola Zannone. Purpose control: Did you process the data for the intended purpose? In Willem Jonker and Milan Petkovic, editors, *Secure Data Management*, volume 6933 of *Lecture Notes in Computer Science*, pages 145–168. Springer, 2011.

[PS07] Hyoungmin Park and Kyuseok Shim. Approximate algorithms for k-anonymity. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD '07, pages 67–78, New York, NY, USA, 2007. ACM.

[PWGB10] Suraj Pandey, Linlin Wu, Siddeswara Mayura Guru, and Rajkumar Buyya. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 400–407. IEEE, 2010.

[PZCG14] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Context aware computing for the internet of things: A survey. *Communications Surveys & Tutorials, IEEE*, 16(1):414–454, 2014.

[RF05] Paolo Remagnino and Gian Luca Foresti. Ambient intelligence: A new multidisciplinary paradigm. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 35(1):1–6, 2005.

[RJD05] Ueli Rutishauser, Josef Joller, and Rodney Douglas. Control and learning of ambience by an intelligent building. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 35(1):121–132, 2005.

[Rou08] Antoinette Rouvroy. Privacy, data protection, and the unprecedented challenges of ambient intelligence. *Studies in Ethics, Law, and Technology*, 2(1), 2008.

[RSMP11] S Raj Rajagopalan, Lalitha Sankar, Soheil Mohajer, and H Vincent Poor. Smart meter privacy: A utility-privacy framework. In *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*, pages 190–195. IEEE, 2011.

[RWLN89] J. Rothenberg, L. E. Widman, K. A. Loparo, and N. R. Nielsen. The nature of modeling. In *Artificial Intelligence, Simulation and Modeling*, pages 75–92, 1989.

[SBMP08] Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. *EMF: eclipse modeling framework*. Pearson Education, 2008.

[Sch85] J David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st international Conference on Genetic Algorithms*, pages 93–100. L. Erlbaum Associates Inc., 1985.

[Sch06] Douglas C Schmidt. Guest editor's introduction: Model-driven engineering. *Computer*, 39(2):0025–31, 2006.

[SHGW05] Wim Schreurs, Mireille Hildebrandt, Mark Gasson, and Kevin Warwick. Report on actual and possible profiling techniques in the field of ambient intelligence. Technical report, FIDIS (Future of Identity in the Information Society), Deliverable D7.3, http://www.fidis.net, 2005.

[Sol06] Daniel J. Solove. A Taxonomy of Privacy. *University of Pennsylvania Law Review*, 154(3):p477+, January 2006.

[ST94] Bill N Schilit and Marvin M Theimer. Disseminating active map information to mobile hosts. *Network, IEEE*, 8(5):22–32, 1994.

[Ste02] Gerhard Steinke. Data privacy approaches from us and eu perspectives. *Telematics and Informatics*, 19(2):193 – 200, 2002. Regulating the Internet: EU and US perspectives.

[Swe02a] Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):571–588, October 2002.

[Swe02b] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness & Knowledge-Based Systems*, 10(5):557, 2002.

[VLA87] Peter JM Van Laarhoven and Emile HL Aarts. Simulated annealing. 1987.

[WAD+05] Jehan Wickramasuriya, Mohanned Alhazzazi, Mahesh Datt, Sharad Mehrotra, and Nalini Venkatasubramanian. Privacy-protecting video surveillance. In *Electronic Imaging 2005*, pages 64–75. International Society for Optics and Photonics, 2005.

[WB90] Samuel D. Warren and Louis D. Brandeis. The right to privacy. *Harvard Law Review*, 4(5):pp. 193–220, 1890.

[Wei91] Mark Weiser. The computer for the 21st century. *Scientific american*, 265(3):94–104, 1991.

[Wei93] Mark Weiser. Hot topics-ubiquitous computing. *Computer*, 26(10):71–72, 1993.

[Wes67] Alan Westin. *Privacy and Freedom*. New Jork Atheneum, 1967.

[WGNF12] Xuetao Wei, Lorenzo Gomez, Iulian Neamtiu, and Michalis Faloutsos. Permission evolution in the android ecosystem. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 31–40. ACM, 2012.

[WHFG92] Roy Want, Andy Hopper, Veronica Falcao, and Jonathan Gibbons. The active badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102, 1992.

[WJ94] Matt P Wand and M Chris Jones. *Kernel smoothing*. Crc Press, 1994.

[WZ11] Wiktoria Wilkowska and Martina Ziefle. Perception of privacy and security for acceptance of e-health technologies: Exploratory analysis for diverse user groups. In *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2011 5th International Conference on*, pages 593–600. IEEE, 2011.

[XT06a] Xiaokui Xiao and Yufei Tao. Anatomy: simple and effective privacy preservation. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 139–150. VLDB Endowment, 2006.

[XT06b] Xiaokui Xiao and Yufei Tao. Personalized privacy preservation. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, pages 229–240, New York, NY, USA, 2006. ACM.

[ZBT07] Eckart Zitzler, Dimo Brockhoff, and Lothar Thiele. The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In *Evolutionary multi-criterion optimization*, pages 862–876. Springer, 2007.