Chapter 6
Collective Sensing Platforms

Martin Atzmueller[1], Martin Becker[2,3], and Juergen Mueller[1,3]

Abstract This chapter provides an overview of web-based information and communications technology platforms that collect and display sensor based information. We focus on collective sensing platforms that allow to extend the collected sensor information, e. g., using tags or other annotations. We provide an overview on such platforms and discuss critical issues such as big data and sensor cloud storage. Furthermore, we discuss specific technological challenges, covering the complete data cycle from the smartphone application to the web system, and its effectiveness.

6.1 Introduction

Collectively organized information like citizen science applications using sensors – as a form of sensor-based crowdsourcing – enable a variety of scientific as well as industrial applications [28] in addition to enhancing our understanding of certain phenomena for the overall benefit of human knowledge and science. The collected data of such applications and its embedded collective intelligence [2, 36, 38] can then be leveraged for enhancing methods in various application contexts, e. g., for recommendations, various resource optimization problems, or for obtaining insights into social interactions, e. g., [31, 40–42], see also Chapter "Observing Human Activity through Sensing" by Gautama et al. With the advent of ubiquitous and mobile computing, many new applications have been designed for mobile devices enabling people to record environmental data (light, noise, etc.) by making use of embedded sensors, such as a microphone, camera, accelerometer, gyroscope, and GPS receiver. Hence, methods and techniques of flexibly acquiring and handling this data play a central role in paving the way towards behavioral shifts within large citizen populations. In this chapter, we provide an overview on collective sensing platforms, and discuss critical issues such as big data processing and sensor cloud storage.

University of Kassel, Research Center for Information System Design · University of Wuerzburg, Data Mining and Information Retrieval Group · University of Hannover, L3S Research Center

This is a preprint of a Chapter in: Loreto et al. (eds.) Participatory Sensing, Opinions and Collective Awareness. Springer Verlag, Berlin/Heidelberg, Germany. In Press

The remainder of this chapter is organized as follows: Section 6.2 provides an overview on aspects concerning collective sensing. After that, we summarize issues of big data processing in Section 6.3, and sensor cloud storage aspects in Section 6.4. Next, we discuss specific platforms in Section 6.5. Finally, Section 6.6 concludes the chapter with a summary and outlook on interesting future directions.

6.2 Overview

In the following, we provide a brief overview on aspects concerning collective sensing. This includes a brief review of the involved topics, as well as platforms. We will discuss these in more detail in the following sections, including the respective platforms. In addition, we discuss issues of big data processing in the context of data analytics, data processing and data management. Furthermore, we discuss important aspects of storage in the context of collective sensing.

Resch [44] defines collective sensing as "analyzing aggregated anonymized data coming from collective networks", including systems like Flickr, Twitter, Foresquare, and the mobile phone network "collective networks". The focus is mainly subjective data created by users such as comments, impressions, or perceptions. Blaschke et al. [23] takes a more general approach and proposes "interoperable, standardized data fusion options" to be the key feature to collective sensing while not being restrictive about the data sources. Similar to Resch, the emphasis is on the ability to create "new information [...] through a combination of individual data-threads". Personal sensing is mentioned as a part of collective sensing. Vuran et al. [47] define collective sensing in the context of wireless sensing networks (WSN), with the main feature of gaining knowledge from collectively gathered information. The term "collective sensing" is also used in robotics with the same connotation [22]. Thus, all definitions of collective sensing share the same underlying principle: combining a possibly large set of data streams from different sensors in order to yield information, which is not extractable from any single data stream. The most general approach does not restrict the data types being combined.

In order to leverage the wide variety of possible data streams, data must be collected, stored, and provided to applications, which aim to extract knowledge from the collected information. To this end, middleware platforms, which centralize the process of storing, processing, and accessing the collected data, such as Xively, ThingSpeak, or Ubicon/EveryAware, have emerged. Such platforms must handle certain layers of the collective sensing process, which include to certain extents: data definition (what kind of data can be accessed or stored), data alignment (store relations between data points, e. g., time, type, etc.), data processing (aggregation and evaluation of data) and data communication (querying and notification flexibility, access rights, visualization). At the data definition layer there are different sensing paradigms to handle, e. g., remote sensing, in situ sensing, stationary sensing, mobile sensing, citizen science as mobile, people driven sensing, densely vs. sparse, high vs. low quality, subjective vs. objective, etc.

All those paradigms must be handled so that the incoming data can be stored optimally for a possibly large set of different processing algorithms, and can be flexibly accessed later. The data alignment layer is closely related to data storage. Some data alignment aspects can be handled by standardized data formats. Other aspects of data alignment must be handled by post-processing the data, which is covered by the processing layer. The processing layer itself includes data alignment, but also prepares data for access and visualization. On top of that, data mining algorithms can be applied to aggregate and evaluate data and extract knowledge, which must also be prepared for easy access. Finally, the access layer provides the interface to access the processed data by applications and individual users. It may also push data to recipients. At this level it is important to correctly handle access rights as well.

6.3 Big Data Aspects

With the emergence of large-scale data collection, e. g., provided by web-based applications, social computing, ubiquitous computing, mobile computing, and collective sensing, the storage, processing and abstraction of big data is one of the current key research topics [24, 32].

In this section, we will focus on big data processing, aggregation, and abstraction aspects. In particular, we focus on the *Lambda* architecture [39] for handling big data, the Map/Reduce framework [25], and other challenging aspects in the context of collective sensing platforms [20]. In the following, we first outline some typical system properties and challenges in Big Data systems, before we briefly summarize the Lambda architecture, and the Map/Reduce framework.

6.3.1 Overview

According to the *four V* criteria [32] (i. e., velocity, volume, variety, and veracity), big data requires efficient methods to handle the rapidly incoming data with appropriate response time (velocity), the large number of data points (volume), many different heterogeneously structured data sources (variety), and data sources with different quality and provenance standards (veracity). Therefore, there are several challenges that have to be addressed, such as the handling of structured and unstructured data, metric vs. qualitative data, information extraction for textual data, as well as integration techniques for the comprehensive set of data sources. For semi-structured data, e. g., rule-based methods [11,33] and expectation-driven approaches can often be successfully applied (e. g., [12, 34]). Possible extensions include techniques for handling unstructured data, and according learning methods.

Modeling large and heterogeneous data in a data-warehouse [48] requires according modeling and indexing techniques. These can be implemented, e. g., using the Map/Reduce framework [25] summarized below.

Before starting with a data processing framework, different questions and requirements need to be clarified, e. g., according to the types, structure and accuracy of data that is to be implemented, which can often be supported using exploratory approaches, e. g., [3]. For subjective data, e. g., adequate validation and introspection methods, e. g., [6, 15, 16] often need to be applied in order to ensure a sufficient data quality in the further big data processing pipeline. Then, also data alignment, aggregation, and analysis requirements need to be defined.

Furthermore, in addition to data processing frameworks, big data as obtained by collective sensing solutions can be turned into *smart data* by the integration of semantic information, cf., [20]. This can also help in determining aspects of data quality, validity and trust, e. g., considering provenance information of the data, see also Chapter "Privacy, Trust and Incentives in Participatory Sensing" in this part of the book, for a discussion on socioeconomic issues. Considering the *four V* criteria discussed above, especially the velocity aspect also requires support for continuous semantic annotation, in order to ensure valid and high quality data.

6.3.2 Lambda Architecture

According to Marz and Warren [39], system properties of a Big data system typically exhibit the following system properties: They should provide a *general* data framework that is *extensible*, enables *ad-hoc queries* with *minimal maintenance*, and *debugging capabilities*. For data storage, this implies mechanisms for handling the *complexity* of data, e. g., for preventing corruption issues and maintenance issues. Further, *robustness and fault-tolerance* should be enforced, as well as *low latency reads and updates*. This also points to *scalability* issues concerning horizontal and vertical scalability, and the option of obtaining *intermediate results* and views, according to some concept of reproducibility.

The lambda architecture incorporates these system principles and especially tackles the concept of reproducibility of results and views for dynamic processing. Essentially, it allows to compute arbitrary functions on arbitrary datasets in real-time [39]. The lambda architecture is structured into several layers briefly summarized in the following:

- Batch layer: continuously (re-)computes batch views using the immutable master data records.
- Serving layer: indexes query view, performs updates, and provides access to the dataset. Only batch updates and random reads are supported, no (distributed) writes.
- Speed layer: high-latency updates; fix batch layer lag; needs fast algorithms for incremental updates.
- Complexity isolation: random writes only need to be supported in speed layer. Results are then merged with the precomputed data from the batch layer.

In the next section, we briefly summarize the Map/Reduce framework, that can be utilized for implementing, e. g., the batch layer.

6.3.3 Map/Reduce

Map/Reduce [25] is a paradigm for scalable distributed processing of big data. Its core ideas are based on the functional programming primitives *map* and *reduce*. Whereas *map* iterates on a certain input sequence of key-value pairs, the *reduce* function collects and processes all values for a certain key. The Map/Reduce paradigm is applicable for a certain computation task, if this task can be divided into *independent* computational tasks, such that there is no required communication between these. Then, large tasks can be split up into subtasks according to a typical divide-and-conquer strategy.

Map/Reduce is a powerful paradigm for processing big data – with a prominent implementation given by the *Hadoop* framework[1] supported by the HDFS filesystem, and big data databases such as Hive[2] and HBase[3]. Map/Reduce tasks can also be utilized for batch processing in the Lambda architecture discussed above, such that continuous views are (re-)computed by the respective Map/Reduce jobs. These batch tasks can then be complemented by tools for distributed realtime computation like the Storm framework[4], or the Flink[5] platform. This allows a comprehensive data processing pipeline for big data in the Lambda architecture, combining realtime together with Map/Reduce techniques. Alternatives to Map/Reduce, especially considering *in-memory computation* with large datasets include, for example, the Spark[6] [50] and Flink platforms.

6.4 Sensor Cloud Storage Aspects

As outlined above, collective sensing usually goes along with a huge amount of collected data that need to be organized and stored in an efficient way. The data are usually of different data types, which increases the complexity, i. e., they exhibit a large *variety* as described above. Also, there are many areas that require continuous information in order to ensure high quality services and/or products, e. g., areas like healthcare, manufacturing, or environmental monitoring. Wireless sensor networks (WSN) provide this continuous information. They consist of distributed

[1] `http://hadoop.apache.org/`
[2] `http://hive.apache.org/`
[3] `http://hbase.apache.org/`
[4] `http://storm.apache.org/`
[5] `http://flink.apache.org/`
[6] `http://spark.apache.org/`

nodes that gather data for a given purpose, creating a huge amount of data that has to be stored and processed. However, they suffer from different disadvantages that are subject of recent research: limited memory, energy, and computation capabilities to name just a few of them [1, 43].

Cloud computing offers virtually unlimited storage, processing power, no energy issues, and more. Therefore, a combination of both, WSN and cloud computing, addresses the previously mentioned issues [27,49]. In this context, relational databases are not capable to handle the data efficiently in the cloud. Therefore, NoSQL databases have emerged, utilizing a hashed key-value storage. NoSQL is able to deal with very large semi-structured data – with the following challenges [29, 30]:

- **High performance:** The data must be quickly accessible, independent of the amount of stored data. Reading and writing must happen in real-time, especially in high concurrency scenarios.
- **Huge storage:** The most basic need is to store all data. This includes large partitions on the one hand and a great extent of distribution on the other hand.
- **High scalability:** The infrastructure has to be able to increase with a growing number of collected data and participants. On the other hand, it should be able to reduce the required infrastructure when the share is decreasing.
- **High availability:** Data should be accessible from everywhere. This enables flexible monitoring and analysis of the data collected so far.
- **Complex queries:** In order to enable complex data analysis, an advanced query language has to be provided. It has to handle multiple tables with lots of data across multiple distributed platforms.
- **Resource optimization:** Freeing sensor nodes from some of their tasks like storing and processing of data, reduces their complexity. This can lead to cheaper sensor nodes, a reduction in power consumption, and a maximization of the networks' life time.
- **Lower management and operational costs:** Proprietary devices are often not able to communicate between different vendors. Collecting all data in a central repository with a standardized protocol can overcome this.

Zheng et al. [51] propose a cloud storage platform for pervasive computing environments. The authors address the limitations of single sensor nodes, as mentioned before, and present an architecture to solve them, mainly focusing on proprietary daily live sensors like smartphones or media players. However, their approach is too narrow to support wider ranges of sensor values like environmental monitoring data. In contrast, a modern sensor cloud storage should support literally every kind of data and provide as flexible access to it as possible. An exemplary generic and highly extensible data model for sensor cloud storage has been implemented, for example, in the context of the EveryAware project described below.

Another important dimension of sensor data storage and access concerns the issue of *privacy*. While this issue is very relevant, it is nevertheless not very prominent throughout the majority of the available frameworks and platforms. A notable exception is the Ubicon software platform which is discussed in the next section. It provides flexible privacy settings for data access, implementing according to guidelines for the socio-technical design of ubiquitous computing systems, cf., [7, 19].

6.5 Collective Sensing Platforms

There are a number of frameworks and toolkits supporting collective sensing on different levels regarding the layers discussed in Section 6.2. In this section, we cover several such platforms. We compare their capabilities and highlight differences. In particular, we focus on the Ubicon [4, 5] software platform for ubiquitous social computing, and the conceptual data model devised for the EveryAware backend built on top of Ubicon [5, 21]. Both are available under an open source license[7].

In the following, we start with a description of Ubicon, and provide and overview on its system architecture. After that, we discuss the conceptual data model used in EveryAware, as an example of a generic and highly extensible data model for sensor data. Finally, we summarize several related platforms and discuss them in context.

6.5.1 Ubicon

The Ubicon software platform [4, 5] aims at enhancing ubiquitous and social networking, as a platform for flexibly implementing applications in that context. It aims at supporting applications at the intersection of ubiquitous and social computing, integrating functionalities of both environments, providing efficient and effective for building applications in areas like ubiquitous and social computing, internet of things, participatory sensing, and social crowd sourcing.

Ubicon provides a number of components for data collection, processing, and serving. At its core, it provides the means for creating and hosting customized applications. Grounded by fundamental principles of big data storage, processing, and analytics [39], Ubicon features flexible ways for adaptions and extensions in the respective applications. Below, we first present the general system architecture, before we describe the specialized conceptual data model implemented for the EveryAware backend. This model provides for easy implementation of collective sensing modules, as exemplified by the EveryAware applications described in the next chapter.

6.5.1.1 System Architecture

Figure 6.1 shows a conceptual overview of the system's architecture. From a data-centric view, Ubicon implements a data storage, processing, and serving pipeline similar to the *lambda architecture* [39] for handling and managing big data. In that way, core concepts such as immutability and recomputation are transparently enabled by the platform. Accordingly, the data flow is organized in the layers *immutable data storage*, *data processing*, and *data serving* providing flexible and transparent access to the data, e. g., for implementing big data analytics using Map/Reduce [25]. The functionality for each of these layers is backed by the Ubi-

[7] `https://bitbucket.org/ubicon/ubicon`

con core which provides canned functionality, i. e., framework classes and interfaces, which can be utilized throughout different applications. For more details on the architecture, see Atzmueller et al. [5].

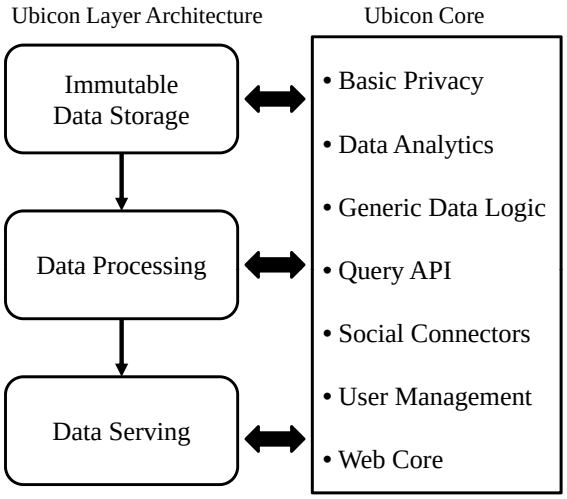

Fig. 6.1 Conceptual overview on the architecture of the Ubicon software platform [5].

Overall, Ubicon enables the observation of physical and social activities. Typically, applications utilize the provided core components, interfaces, and classes and extend the overall workflow according to their individual application requirements, as also described in the chapter "Applications for Environmental Sensing in EveryAware" by Atzmueller et al. in this part of the book, for the applications of the EveryAware web application backend built on top of Ubicon. Other applications include, for example, the Conferator [8], a social conference guidance system, and MyGroup, an application for enhancing social interaction in working groups. Both use active SocioPattern[8] RFID tags, which allow to localize participants and to collect their face-to-face contacts. This allows for highly personalized profiles in the systems, which can be applied, e. g., for community mining and for generating recommendations or notifications. The tags allow the coupling of real world (offline) data, i. e., face-to-face contacts, with the online social world, e. g., given by online interactions within the system or in linked online social networks. Using collective sensing data obtained using the Conferator system, Atzmueller et al. [9] analyze the interactions and dynamics of the behavior of participants at conferences; similarly, the connection between research interests, roles and academic jobs of conference attendees is further analyzed in Macek et al. [37]. Connecting collective sensing data to online data, Scholz et al. [46] analyzed the predictability of links in face-to-face contact networks and additional factors also including online networks.

[8] `http://sociopatterns.org/`

There are several related frameworks and platforms, e. g., concerning context-awareness. The Context Toolkit [26, 45], for example, provides a conceptual framework for the rapid development of context-aware applications. Similarly, Bannach et al. [17, 18, 35] present the context recognition network toolkit/toolchain for building context-aware pervasive applications.

Compared to these toolkits, Ubicon focuses at supporting applications that consider *both* ubiquitous and social aspects. In addition, Ubicon is no general toolkit for rapid prototyping, but aims at providing general framework support for implementing and hosting ubiquitous and social applications in high-availability online scenarios. This is achieved by providing a layered template architecture with an efficient and effective data storage and processing chain. Then, applications implement this template using the modules provided by the Ubicon core components. In addition, applications can also make use of the same platform components, such that they are hosted on the same server for potentially sharing data and providing an integrated user experience across applications.

6.5.1.2 Conceptual Data Model used in EveryAware

The EveryAware project facilitates the combination of sensor and subjective data, i.e., sensor measurements like noise or air quality related recordings and impressions, perceptions and social context. The platform enables users to collect and visualize environmental information and at the same time augment the collected data with arbitrary information explicitly supporting subjective context.

According to these requirements, we designed a specialized extensible data model. In the following, we first introduce the core data model which enables the combination of subjective and objective data and then give a short introduction into the EveryAware access and visibility concepts.

Core Data Model

The Ubicon framework provides several building blocks to support collective sensing that can be embedded into its generic data storage, processing, and serving pipeline as introduced above. In addition, it implements structures for user management and privacy handling. In order to support arbitrary sensor data, a specialized data model has been defined in the context of the EveryAware project. This also especially addresses the integration of subjective data, such as user perceptions or tags, which was one of the goals of the EveryAware project. Thus, the conceptual layer of EveryAware defines corresponding basic entities and features.

Core concepts are data points (with descriptions), sessions, and feeds. Data points and sessions can be extended by other data points. Each data point consists of a set of fixed description attributes in addition to the actual data. These attributes ensure the processability as well as dynamic querying of arbitrary content. The description attributes are divided into three categories:

- Meta attributes are attributes which allow to keep track of data independent information like received time, recording time, device ID, or session ID.
- Geo attributes make it possible to record the location of the sample being taken including longitude and latitude as well as accuracy and the provider of the location fix.
- Content attributes describe the content and its format. They help the system to further process the data. These attributes include the data type (e. g., air, noise) and format (e. g., JSON, XML, PNG).

Based on these attributes it is possible to define a variety of concepts for augmenting the data and provide subjective or social context. These concepts are sessions, extensions and feeds as listed below.

- *Sessions* are collections of data points limited to a fixed timespan. Sessions allow to introduce semantic entities such as "my way to work" or "a stroll in the park".
- Using *extensions*, data points as well as sessions can be extended with additional information using other data points. This makes the data representation very flexible and inherently supports the augmentation of objective data with a semantic context. One application is tagging. Sessions and data points can be tagged by extending them using tag data points referring to the respective data point or session IDs to be tagged. Tagging is not only restricted to actual text-tags but can be any kind of data including videos, sound files, or air quality measurement. Using this scheme, it is also possible to update data points as well as sessions after they have been sent without losing the original data. Since no raw data is deleted, this also allows to always access the version history of a data point.
- *Feeds* can be used for organizing data points. A data point is always part of the global feed, but can also be pushed into several other feeds. Users can contribute to existing feeds or create their own ones. While useful for organizing data points, feeds also allow to attach data points to real world entities such as major events like music festivals, places like the Eiffel Tower, or portable things like a smartphone. Feeds can be access restricted and a visibility level can be specified for each data point in a feed.

Access & Visibility

As discussed above, privacy is a major concern for users of data collection platforms. Therefore, adequate mechanisms and structures for privacy with respect to storage, processing and data serving need to be implemented. From an application perspective, this usually relates specifically to *data access* and *visibility*. Therefore, we aim at providing access and visibility concepts that give the users fine-grained control of what they want to share with others. To this end we base our model on feed-wise access and further define visibility levels within feeds allowing the user to choose if data can be accessed in detail or only aggregated with other data.

In general, feeds can be *open* or *closed* concerning read and write access, where *write access* refers to the possibility of adding new data points to a feed. Open feeds are accessible by everyone including anonymous users. Closed feeds are only accessible by a limited set of users (i. e., *members*). The access restriction allows users to create feeds and share them with friends or other interested users without making their data publicly available.

Since users may want to contribute in different ways to the data collected within feeds and corresponding statistics which might be derived from it, the EveryAware concept introduces visibility levels for each data point in a feed.

There are four visibility levels:

1. *Details* allows everyone who has access to the feed to see the raw content as well as the description attributes of the data point.
2. *Statistics* restricts the data point to be considered in user statistics derived from the data points in the feed, e. g., average values for the corresponding user.
3. *Anonymous* restricts the data point to only be considered in overall statistics derived from the data points in the feed, e. g., average values for an area or timespan. No association with the user is possible.
4. *None* allows only the owner of the data point to access the data point and its description attributes.

This scheme was introduced in order to allow users to share their data even if they are concerned about single data points or user specific statistics being shared.

6.5.2 Other Platforms

After introducing Ubicon and EveryAware in detail, we now summarize characteristics and features of several other platforms which support collective sensing. We especially focus on the different data models and querying capabilities. Note that regarding privacy, none of the systems includes access and visibility settings as proposed by the EveryAware backend. Most other systems group their collected data into feeds which can then be made accessible to other users, directly or by access key. In this regard, EveryAware provides a more flexible way of granting access to the shared data if correctly implemented. However, even EveryAware currently does not provide explicit support for post-processing or anonymizing location data.

In the remainder of this section, we first outline the Xively platform. Afterwards, we compare the other platforms with regard to corresponding similarities and differences.

6.5.2.1 Xively

On the data definition level, Xively [9] defines data points as pairs of timestamps and values. Values can be any kind of textual input. Data points are grouped into data streams, where a data stream usually represents a sensor or "channel" on a device. Data streams are defined by a name, tags, the unit of the values, and a symbol describing the unit. Tags are used for searching data streams. While unit and symbol restrict the input of the data stream on a semantic level, they are not enforced when uploading data. Data streams are then grouped together into feeds. Feeds usually represent devices which are made of several sensors, e. g., a sensor box such as the Air Quality Egg[10]. Each feed may have one location stream. Feeds may also have several meta attributes like tags, a description, a website, or an associated e-mail address. Based on their definition of data streams Xively offers visualization capabilities for numeric streams. The definition of data streams is limited in a sense that data alignment or coupling between streams of a device can only be achieved based on timestamps, i. e., data streams are semantically independent.

Complex data types, as for example, accelerometer data consisting of a triple of values, either need to be modelled as several data streams or by defining a string representation used to set the value of the stream. The latter will break the visualization capabilities and limit the possible set of queries. Xively does not offer any advanced data processing besides calculating basic statistics of numeric data streams. On the data communication layer, Xively offers several endpoints for querying data. Query parameters include feeds, streams, and time intervals. It also defines a trigger API which enables to push messages upon certain events, like receiving a new value or when a value grows above a certain level. For device management, Xively features mechanisms to efficiently deploy and manage batches of devices including access restrictions based on API keys.

6.5.2.2 ThingSpeak

ThingSpeak[11] has the same basic structure as Xively. The vocabulary is a little different: ThingSpeak defines channels (Xively: feeds), feeds (Xively: data streams), and events (Xively: data points) as basic building blocks. We will use the Xively terminology for clarity reasons. For ThingSpeak feeds are limited to eight data streams extended by a location stream as well as a 140 characters long status stream. Each data point actually is a tuple of eight values, one for each data stream, as well as a location and a status message. This also allows to retrieve the tuple as a unit. Thus, in contrast to Xively, data streams are not independent making data alignment easier between data streams of the same feed. There is no additional meta-data like units or tags attached to data streams.

---

[9] `https://xively.com/`

[10] `http://airqualityegg.com/`

[11] `https://thingspeak.com/`

On the feed level several meta-attributes are available including a description, tags, a URL, and a video. The missing tags on the stream level complicate the search for data streams. Also, due to missing units as well as missing tags on the stream level, additional knowledge is required when comparing individual data streams. Just like in Xively data processing is limited. Querying data is based on time intervals. Additionally, numeric values can be constrained by upper and lower bounds and values can be summarized using different statistics like average, sum, or median. In addition, ThingSpeak allows to push data via Twitter or HTTP requests when a data stream reaches a certain status.

6.5.2.3 Open.Sen.se

Open.Sen.se[12] is very similar to Xively in how it organizes data. Just like in ThingSpeak the vocabulary is slightly different but the semantics are the same: Open.Sen.se defines devices (Xively: feeds), feeds (Xively: data streams), and events (Xively: data points) as basic building blocks. Again we will be using Xively's terminology. Just like Xively a feed may contain several data streams where data points from the same "sensor" are collected. As in Xively the streams are independent. No explicit location stream is defined. Thus, when uploading a location (or any tuple based data type), longitude and latitude must either be posted in different streams and those streams must be aligned using the timestamp, or the location must be posted as a custom character string (the documentation only shows numeric data values as input).

The Open.Sen.se API[13] does not allow to add tags or any descriptive content to feeds or streams which makes collected data less understandable. Data points can specify a unit, but this is not enforced. The request API allows to retrieve by data feed or for each data stream separately. Simple constraints can be specified when accessing the data, like "greater than", "lesser than" or "equals".

6.5.2.4 Exosite Portals

Exosite Portals[14] is devided into two components: the One Platform[15] which is the backend used by the Portals component as backend. Portals is a web based API managing One Platform resources. Both use different vocabulary, but in general the structure consists of data ports (Xively: data streams) and data points. Data ports are grouped together by clients which are not really an equivalent of feeds in Xively since they are missing specific location streams and additional meta data. Data points can be numeric or character strings. Binary data is also supposed to be

[12] http://open.sen.se, accessed on 19.02.2014

[13] http://open.sen.se/dev/, accessed on 19.02.2014

[14] https://exosite.com/, accessed on 19.02.2014

[15] http://support.exosite.com/hc/en-us/articles/200397956, accessed on 19.02.2014

supported but this is not documented in the API[16]. Data streams are independent and no explicit location stream is defined. Thus, as in Open.Sen.se, locations (just like other tuple based data types) must be emulated.

There are two further concepts in Exosite Portals which are worth mentioning: client hierarchies and data processing. As mentioned before clients in Exosite Portals do not match feeds. Clients can contain any data port and data ports can be part of any client. Exosite Portals can build hierarchies of clients. This feature is used mainly for access restrictions. Furthermore Exosite Portals explicitly supports data processing. When defining data ports it already allows to modify incoming data in the fly, by using functions like module, addition, etc.. Further more it allows to write custom scripts in the Lua scripting language and store results in new data ports. When querying data from Exosite Portals a single data port is accessed. Classically, time intervals constraints are supported. Additionally simple downsampling is supported. Exosite Portals also provides a powerful events and alert API enabling to push data triggered by a large variety of triggers. Custom triggers are supported.

6.5.2.5 Other

There are other platforms taking similar approachs as Xively, ThingSpeak, etc.. In the following, we provide more examples and sketch the main differences.

- SensorCloud[17] focuses on an efficient binary data protocol. The website also states efficient visualizations and custom analysis using scripting languages like Octave.
- Device Cloud[18] also allows to manage firmware updates of devices and focuses on large sensor deployments and their maintenance.
- Eye on Earth[19] takes a different approach, focusing on letting users create and share custom maps.
- OpenIoT[20][21] is not a platform itself but a project focused on providing a complete toolchain for internet of things deployments.
- Fulcum[22] and EpiCollect are more focused on forms submitted by users than on sensor data and the internet of things aspect which is a key theme in the collective sensing aproach.

---

[16] https://github.com/exosite/api/tree/master/rpc#identifying-resources, accessed on 19.02.2014

[17] http://www.sensorcloud.com/

[18] http://www.etherios.com/products/devicecloud/

[19] http://www.eyeonearth.org/

[20] http://openiot.eu/

[21] https://github.com/OpenIotOrg/openiot/wiki/OpenIoT-Architecture

[22] https://web.fulcrumapp.com

6.6 Conclusions

In this chapter, we have outlined several dimensions and specific aspects of collective sensing systems including an overview on definitions of collective sensing and state-of-the-art platforms. Furthermore, we discussed critical dimensions in this context, i. e., aspects of big data analytics, processing, and management, as well as sensor cloud storage. We discussed these issues in detail considering the server applications, in particular the Ubicon software platform, cf., [4, 5], applied in the EveryAware research project.

Overall, the presented systems mostly focus on data generated by sensors. Notable exceptions are given by Ubicon, and the applications built on top of it, respectively. For example, the EveryAware backend explicitly supports to augment data, in particular for including subjective information like user perceptions or tags, and supports it using a highly extensible data model for sensor data. Further examples are the Conferator and MyGroup applications implemented using Ubicon which allow the annotation of (abstracted) sensor data [7].

These subjective and user driven aspects are very important in order to gain deeper understanding of the processes and environments generating the data. For example, an unexpected high value of temperature carries more value if a user also provides the cause of such a measurement, the respective event, and its context. Thus, the collected data can only be fully leveraged if information is collected which allows to derive the data's context and how it is to be interpreted. Future platforms should directly support collecting such meta data and explicitly include user feedback. In addition, these platforms should further try to interpret user feedback and extract joint information from the combination of subjective and objective data, also using exploratory tools and methods for getting first insights into the data, e. g., [10, 13, 14].

References

1. Alamri, A., Ansari, W.S., Hassan, M.M., Hossain, M.S., Alelaiwi, A., Hossain, M.A.: A survey on sensor-cloud: Architecture, applications, and approaches. International Journal of Distributed Sensor Networks (2013). DOI 10.1155/2013/917923
2. Atzmueller, M.: Onto Collective Intelligence in Social Media: Exemplary Applications and Perspectives. In: Proc. 3rd International Workshop on Modeling Social Media (MSM 2012), Hypertext 2012. ACM Press, New York, NY, USA (2012)
3. Atzmueller, M.: Subgroup Discovery – Advanced Review. WIREs: Data Mining and Knowledge Discovery **5**(1), 35–49 (2015). DOI 10.1002/widm.1144
4. Atzmueller, M., Becker, M., Doerfel, S., Kibanov, M., Hotho, A., Macek, B.E., Mitzlaff, F., Mueller, J., Scholz, C., Stumme, G.: Ubicon: Observing social and physical activities. In: IEEE International Conference on Cyber, Physical and Social Computing, CPSCom 2012, Besancon, France, 20-23 November, 2012, pp. 317–324. IEEE, Washington, DC, USA (2012). DOI 10.1109/GreenCom.2012.75
5. Atzmueller, M., Becker, M., Kibanov, M., Scholz, C., Doerfel, S., Hotho, A., Macek, B.E., Mitzlaff, F., Mueller, J., Stumme, G.: Ubicon and its Applications for Ubiquitous Social

Computing. New Review of Hypermedia and Multimedia **20**(1), 53–77 (2014). DOI 10.1080/13614568.2013.873488
6. Atzmueller, M., Beer, S., Puppe, F.: A Data Warehouse-Based Approach for Quality Management, Evaluation and Analysis of Intelligent Systems using Subgroup Mining. In: Proc. 22nd International Florida Artificial Intelligence Research Society Conference (FLAIRS), pp. 402–407. AAAI Press, Palo Alto, CA, USA (2009)
7. Atzmueller, M., Behrenbruch, K., Hoffmann, A., Kibanov, M., Macek, B.E., Scholz, C., Skistims, H., Söllner, M., Stumme, G.: Socio-technical Design of Ubiquitous Computing Systems, chap. Connect-U: A System for Enhancing Social Networking. Springer Verlag, Heidelberg, Germany (2014)
8. Atzmueller, M., Benz, D., Doerfel, S., Hotho, A., Jäschke, R., Macek, B.E., Mitzlaff, F., Scholz, C., Stumme, G.: Enhancing social interactions at conferences. it – Information Technology **53**(3), 101–107 (2011)
9. Atzmueller, M., Doerfel, S., Hotho, A., Mitzlaff, F., Stumme, G.: Face-to-face contacts at a conference: Dynamics of communities and roles. In: Modeling and Mining Ubiquitous Social Media. International Workshops MSM 2011, Boston, MA, USA, October 9, 2011, and MUSE 2011, Athens, Greece, September 5, 2011, Revised Selected Papers, *Lecture Notes in Computer Science*, vol. 7472, pp. 21–39. Springer, Berlin / Heidelberg, Germany (2012). DOI 10.1007/978-3-642-33684-3_2
10. Atzmueller, M., Doerfel, S., Mitzlaff, F.: Description-Oriented Community Detection using Exhaustive Subgroup Discovery. Information Sciences **329**, 965–984 (2016)
11. Atzmueller, M., Kluegl, P., Puppe, F.: Rule-based information extraction for structured data acquisition using textmarker. In: Lernen, Wissensentdeckung und Adaptivität, LWA 2008, Würzburg, Germany – October 06-08, 2008. Proceedings. University of Würzburg, Würzburg, Germany (2008)
12. Atzmueller, M., Lemmerich, F.: VIKAMINE - Open-Source Subgroup Discovery, Pattern Mining, and Analytics. In: Proc. European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases. Springer, Heidelberg, Germany (2012)
13. Atzmueller, M., Lemmerich, F., Krause, B., Hotho, A.: Who are the Spammers? Understandable Local Patterns for Concept Description. In: Proc. 7th Conference on Computer Methods and Systems (2009)
14. Atzmueller, M., Mueller, J., Becker, M.: Mining, Modeling and Recommending 'Things' in Social Media, chap. Exploratory Subgroup Analytics on Ubiquitous Data. No. 8940 in LNAI. Springer, Heidelberg, Germany (2015)
15. Atzmueller, M., Puppe, F.: A Case-Based Approach for Characterization and Analysis of Subgroup Patterns. Journal of Applied Intelligence **28**(3), 210–221 (2008)
16. Atzmueller, M., Puppe, F., Buscher, H.P.: Profiling Examiners using Intelligent Subgroup Mining. In: Proc. 10th International Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP-2005), pp. 46–51. Aberdeen, Scotland (2005)
17. Bannach, D., Amft, O., Lukowicz, P.: Rapid prototyping of activity recognition applications. IEEE Pervasive Computing **7**(2), 22–31 (2008). DOI 10.1109/MPRV.2008.36
18. Bannach, D., Kunze, K.S., Weppner, J., Lukowicz, P.: Integrated tool chain for recording and handling large, multimodal context recognition data sets. In: 12th ACM International Conference Adjunct Papers on Ubiquitous Computing, Ubicomp 2010, Copenhagen, Denmark – September 26-29, 2010. Proceedings, pp. 357–358. ACM, New York, NY, USA (2010). DOI 10.1145/1864431.1864434
19. Baraki, H., Geihs, K., Hoffmann, A., Voigtmann, C., Kniewel, R., Macek, B.E., Zirfas, J.: Towards Interdisciplinary Design Patterns for Ubiquitous Computing Applications. Tech. rep., Research Center for Information System Design (ITeG), University of Kassel, Germany (2014)
20. Barnaghi, P., Sheth, A., Henson, C.: From data to actionable knowledge: Big data challenges in the web of things. Intelligent Systems **28**(6), 6–11 (2013). DOI 10.1109/MIS.2013.142
21. Becker, M., Mueller, J., Hotho, A., Stumme, G.: A generic platform for ubiquitous and subjective data. In: 2013 ACM International Joint Conference on Pervasive and Ubiquitous Comput-

ing , UbiComp 2013; 1st International Workshop on Pervasive Urban Crowdsensing Architecture and Applications, PUCAA 2013, Zurich, Switzerland – September 8-12, 2013. Proceedings, pp. 1175–1182. ACM, New York, NY, USA (2013). DOI 10.1145/2494091.2499776
22. Bishop, J., Klavins, E.: Collective sensing with self-organizing robots. In: 45th IEEE Conference on Decision and Control, CDC 2006, San Diego, CA, USA – December 13-15, 2006. Proceedings, pp. 4175–4181. IEEE, New York, NY, USA (2006). DOI 10.1109/CDC.2006.377102
23. Blaschke, T., Hay, G.J., Weng, Q., Resch, B.: Collective sensing: Integrating geospatial technologies to understand urban systems–an overview. Remote Sensing **3**(8), 1743–1776 (2011). DOI 10.3390/rs3081743
24. Cuzzocrea, A., Song, I.Y., Davis, K.C.: Analytics over large-scale multidimensional data: The big data revolution! In: 14th International Workshop on Data Warehousing and OLAP at 20th International Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom – October 24-28, 2011. Proceedings, pp. 101–104. ACM, New York, NY, USA (2011). DOI 10.1145/2064676.2064695
25. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. Communications of the ACM **51**(1), 107–113 (2008). DOI 10.1145/1327452.1327492
26. Dey, A.K., Abowd, G.D., Salber, D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction **16**(2), 97–166 (2001). DOI 10.1207/S15327051HCI16234_02
27. Foster, I.T., Zhao, Y., Raicu, I., Lu, S.: Cloud computing and grid computing 360-degree compared. In: Grid Computing Environments Workshop at IEEE International Conference for High Performance Cpmputing, Networking, Storage and Analysis, GCE 2008, Austin, TX, USA – November 12-16, 2008. Proceedings. IEEE, New York, NY, USA (2009). DOI 10.1109/GCE.2008.4738445
28. Haklay, M.: Citizen science and volunteered geographic information: Overview and typology of participation. In: Crowdsourcing Geographic Knowledge: Volunteered Geographic Information (VGI) in Theory and Practice, pp. 105–122. Springer, Netherlands (2013). DOI 10.1007/978-94-007-4587-2_7
29. Han, J., E, H., Le, G., Du, J.: Survey on nosql database. In: 6th International Conference on Pervasive Computing and Applications, ICPCA 2011, Port Elizabeth, South Africa, October 26-28, 2011. Proceedings, pp. 363–366. IEEE, New York, NY, USA (2011). DOI 10.1109/ICPCA.2011.6106531
30. Han, J., Song, M., Song, J.: A novel solution of distributed memory nosql database for cloud computing. In: IEEE/ACIS 10th International Conference on Computer and Information Science, ICIS 2011, Sanya, China – May 16-18, 2011. Proceedings, pp. 351–355. IEEE, New York, NY, USA (2011). DOI 10.1109/ICIS.2011.61
31. Kibanov, M., Atzmueller, M., Scholz, C., Stumme, G.: Temporal Evolution of Contacts and Communities in Networks of Face-to-Face Human Interactions. Science China **57** (2014)
32. Klein, D., Tran-Gia, P., Hartmann, M.: Big data. Informatik-Spektrum **36**(3), 319–323 (2013). DOI 10.1007/s00287-013-0702-3
33. Kluegl, P., Atzmueller, M., Puppe, F.: Meta-level information extraction. In: KI 2009: Advances in Artificial Intelligence. 32nd Annual German Conference on AI, Paderborn, Germany, September 2009. Proceedings, *Lecture Notes in Computer Science*, vol. 5803, pp. 233–240. Springer, Berlin / Heidelberg, Germany (2009). DOI 10.1007/978-3-642-04617-9_30
34. Klügl, P., Toepfer, M., Lemmerich, F., Hotho, A., Puppe, F.: Collective Information Extraction with Context-Specific Consistencies. In: Proc. ECML/PKDD, pp. 728–743 (2012)
35. Kunze, K., Bannach, D.: Towards dynamically configurable context recognition systems. In: Activity Context Representation Workshops at the 26th AAAI Conference on Artificial Intelligence, AAAI 2012, Toronto, Canada – July 22-23, 2012. Proceedings. AAAI, Palo Alto, CA, USA (2012)
36. Leimeister, J.M.M.: Collective intelligence. Business and Information Systems Engineering **2**(4), 245–248 (2010). DOI 10.1007/s12599-010-0114-8
37. Macek, B.E., Scholz, C., Atzmueller, M., Stumme, G.: Anatomy of a conference. In: 23rd ACM Conference on Hypertext and Social Media, HT 2012, Milwaukee, WI, USA – June

25-28, 2012. Proceedings, pp. 245–254. ACM, New York, NY, USA (2012). DOI 10.1145/2309996.2310038
38. Malone, T.W., Laubacher, R., Dellarocas, C.: The collective intelligence genome. Spring **51**(3), 21–31 (2010)
39. Marz, N., Warren, J.: Big Data: Principles and Best Practices of Scalable Realtime Data Systems, 1. edn. Manning, Shelter Island, NY, USA (2013)
40. Mitzlaff, F., Atzmueller, M., Benz, D., Hotho, A., Stumme, G.: Community assessment using evidence networks. In: Analysis of Social Media and Ubiquitous Data. International Workshops MSM 2010, Toronto, Canada, June 13, 2010, and MUSE 2010, Barcelona, Spain, September 20, 2010, Revised Selected Papers, *Lecture Notes in Computer Science*, vol. 6904, pp. 79–98. Springer, Heidelberg, Germany (2011). DOI 10.1007/978-3-642-23599-3_5
41. Mitzlaff, F., Atzmueller, M., Benz, D., Hotho, A., Stumme, G.: User-relatedness and community structure in social interaction networks. CoRR **abs/1309.3888** (2013)
42. Mitzlaff, F., Atzmueller, M., Stumme, G., Hotho, A.: Semantics of user interaction in social media. In: Complex Networks IV. Proceedings of the 4th Workshop on Complex Networks CompleNet 2013, *Studies in Computational Intelligence*, vol. 476, pp. 13–25. Springer, Berlin / Heidelberg, Germany (2013). DOI 10.1007/978-3-642-36844-8_2
43. Ponmagal, R.S., Raja, J.: An extensible cloud architecture model for heterogeneous sensor services. International Journal of Computer Science and Information Security **9**(1), 147–155 (2011)
44. Resch, B.: People as sensors and collective sensing-contextual observations complementing geo-sensor network measurements. In: Progress in Location-Based Services, Lecture Notes in Geoinformation and Cartography, pp. 391–406. Springer, Berlin / Heidelberg, Germany (2013). DOI 10.1007/978-3-642-34203-5_22
45. Salber, D., Dey, A.K., Abowd, G.D.: The context toolkit: Aiding the development of context-enabled applications. In: SIGCHI Conference on Human Factors in Computing Systems, CHI 1999, Pittsburgh, PA, USA – May 15-20, 1999. Proceedings, pp. 434–441. ACM, New York, NY, USA (1999). DOI 10.1145/302979.303126
46. Scholz, C., Atzmueller, M., Barrat, A., Cattuto, C., Stumme, G.: New insights and methods for predicting face-to-face contacts. In: 7th International AAAI Conference on Weblogs and Social Media, ICAPS 2013, Cambridge, MA, USA – July 8-10, 2013. Proceedings. AAAI, Palo Alto, CA, USA (2013)
47. Vuran, M.C., Akan, Ö.B., Akyildiz, I.F.: Spatio-temporal correlation: Theory and applications for wireless sensor networks. Computer Networks **45**(3), 245–259 (2004). DOI 10.1016/j.comnet.2004.03.007
48. Witten, I.H., Moffat, A., Bell, T.C.: Managing Gigabytes Compressing and Indexing Documents and Images, 2. edn. Morgan Kaufmann, Burlington, MA, USA (1999)
49. Yuriyama, M., Kushida, T.: Sensor-cloud infrastructure - physical sensor management with virtualized sensors on cloud computing. In: 13th International Conference on Network-Based Information Systems, NBiS 2010, Takayama, Japan – September 14-16, 2010. Proceedings, pp. 1–8. IEEE, New York, NY, USA (2010). DOI 10.1109/NBiS.2010.32
50. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: Cluster Computing with Working Sets. In: Proc. USENIX Conference on Hot Topics in Cloud Computing, HotCloud'10, pp. 10–10. USENIX Association, Berkeley, CA, USA (2010)
51. Zheng, W., Xu, P., Huang, X., Wu, N.: Design a cloud storage platform for pervasive computing environments. Cluster Computing **13**(2), 141–151 (2010). DOI 10.1007/s10586-009-0111-1