Metafier - a Tool for Annotating and Structuring Building Metadata

Emil Holmegaard, Aslak Johansen and Mikkel Baun Kjærgaard
SDU - Center for Energy Informatics
Mærsk McKinney Møller Institute
University of Southern Denmark
Email: {em,asjo,mbkj}@mmmi.sdu.dk

***Abstract*—Buildings often have several systems installed (i.e. different forms of Building Management Systems (BMS)), with the combined goal of providing a good comfort level for the occupants of the building. With such a goal, it would be natural for such systems to share information and collaborate in achieving this goal, but often they work as silos. Improving at scale the energy performance of buildings depends on applications breaking these silos and being portable among buildings. To enable portable building applications, the building instrumentation should be supported by a metadata layer, describing the instrumentation of the building. We have created Metafier, a tool for annotating and structuring metadata for buildings. Metafier optimizes the workflow of establishing metadata for buildings by enabling a human-in-the-loop to validate, search and group points. We have evaluated Metafier for two buildings, with different sizes, locations, ages and purposes. The evaluation was performed as a user test with three subjects with different backgrounds. The evaluation results indicates that the tool enabled the users to validate, search and group points while annotating metadata. One challenge is to get users to understand the concept of metadata for the tool to be useable. Based on our evaluation, we have listed guidelines for creating a tool for annotating building metadata.**

I. Introduction

An important challenge for improving the sustainability of our society is the energy performance of buildings. In this paper, we focus on commercial buildings. Buildings often have several forms of Building Management Systems (BMS), with the combined goal of providing a good comfort level for the occupants of the building. With such a goal, it would be natural that systems share information and collaborate, but often they work as silos [10]. Improving at scale the energy performance of buildings requires software applications that break these silos and are portable among buildings [9].

For a building application to be portable it needs the ability to lookup relevant points as well as metadata annotating these with information relevant to the application in terms of location, type and relationships. The metadata and data exposed by a particular building should have to comply with a set of standard requirements. The standards could require a format for room locations, point[1] types and how points relates to e.g. the HVAC system. For a particular application this set spans the specific metadata used to decouple it from a particular building. We can compare portable building applications to smartphone applications which use the sensing infrastructure in a smartphone, where the different smartphone models expose the different sensors in a more or less common way. The abstraction layer provided by the smartphone OS makes it tractable to develop applications and share them via an app store. The availability of an OS and metadata layer in the buildings space could form the foundation for establishing a similar app store. One, through which a facility manager could upgrade part of the building control system, ideally by a few clicks.

Several Building Operating Systems (BOS) have been proposed including BOSS[5] among others [15, 18, 22]. The building operating systems makes it practical and advantageous to share information between systems and networked devices by adding an abstraction layer. To provide this abstraction layer, there is a requirement for metadata describing the underlying building instrumentation. The metadata should provide information, which can be used for creation of the abstractions. Researchers have focused this for the last decade, and introduced systems for integration of multiple subsystems into a simpler interface. The difficulties for BOS, is the installation phase, where you need to setup all subsystems, devices and annotate required metadata.

Metadata for building infrastructures is often missing, incomplete or structured in ways that make them difficult to use in building applications. When building infrastructure provides metadata, the structure is decided by the vendor of that instrumentation. This can be cumbersome in buildings with multiple vendors, but even more cumbersome working with multiple buildings. The last couple of years, the interest in metadata for buildings have increased, for enabling portable building applications. Bhattacharya et al. [3] analyzed the most common metadata formats, and concluded that none of the formats would be expressive enough to support the building applications described in the literature. Initiatives like Brick [2] and Haystack [17] attempt to define common formats for building metadata.

Metadata for BOS can be generated automatically, with a framework like Zodiac [1]. Zodiac combines classification of raw sensor readings with metadata extracted from the vendor tags of a traditional BMS. However, the success of

[1]A *point* represents a connection between the cyber and physical world which may be discretized into a time series. Such time series contains either sensor readings or actuation requests depending on direction.

978-1-5386-0435-9/17/$31.00 ©2017 IEEE

the extracting metadata from tags depends on the care taken by the specific vendor when naming them. Zodiac does not set a location for the point. This is a key missing piece, if we would like to have applications controlling different parts of the building. Using an automated approach for generating metadata on relationships will produce unforeseen side-effects when the source data contains errors. Such relationships can be easily discovered in a more robust way by involving a facility manager with expert knowledge of the building.

Balaji et al. [2] have created Brick, a metadata format which can hold relationships between points, systems and other elements in a building. Brick stores these relations as subject $\times$ predicate $\times$ object triples. The work has demonstrated how the format support a set of relevant applications. Brick so far does not include tools for annotating metadata, including versioning of changes to the Brick information. However, Brick pushes the metadata community to have one common metadata format, which can store all relationships and important metadata for buildings to enable portable building applications.

We introduce Metafier, a tool for annotating and structuring metadata for building instrumentations. The tool enables humans to efficiently annotate points with metadata. In particular Metafier enables:

- Search for points
- Grouping of points
- Validation of points
- Versioning of metadata
- Data visualization while editing metadata
- Profiles to ensure a consistent level of metadata

In this paper we report a preliminary study of using Metafier for two buildings, with different size, location, age and purpose. We have used profiles, to create a common metadata structure to enable portable building applications, which can be run on the two buildings, without any modifications.

II. The Tool Metafier

Metafier is a tool for annotating and structuring building instrumentation metadata. Metafier does not store any sensor readings, but has an interface used for channel sensor readings. The interface is for the simple Measurement and Actuation Profile (sMAP) proposed by Dawson-Haggerty et al. [4]. sMAP has a REST API for interacting with two databases, one for time series data, and one for associated metadata. sMAP uses a key-value store for metadata. Metafier retrieves time series data and stores annotations for points in the metadata database.

Fig. 1 illustrates the list view in Metafier with all points and groups. New groups can be created by either a search or by clicking on points, Metafier will asks for a name of the group.

Fig. 2 illustrates the edit view of a point, but it would look no different for a group. As shown, the end user can chose multiple profiles for the point.

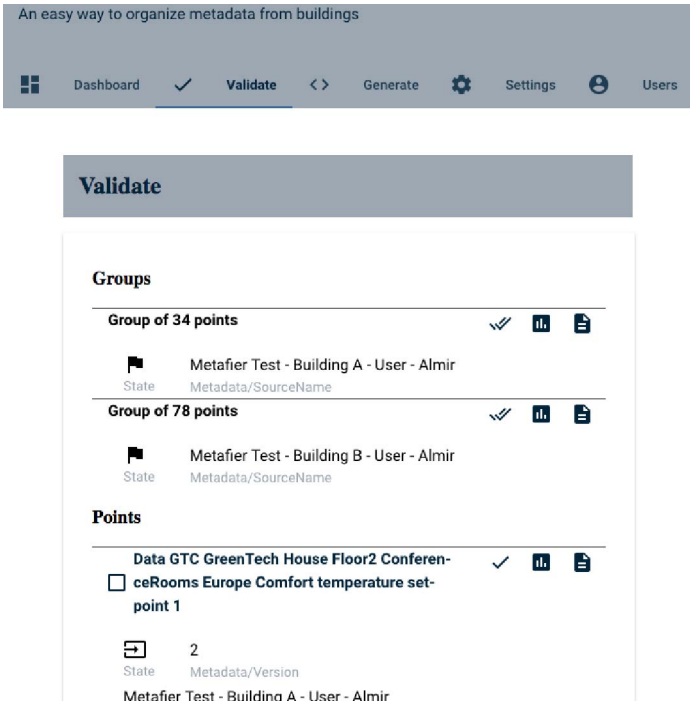

Fig. 1: Screenshot for "list view" with a overview of all points in Metafier

A. Features

We have in Metafier implemented six key features, which are: *search* for a point, *grouping* of points, *visualization* of data from points, *data validation* to verified that the data stream of a point seem to match its annotated metadata, *versioning* of metadata and *metadata validation* to ensure the metadata follows the format specified in profiles. These features have been chosen for reducing the task of annotating building metadata.

We have implemented a search mechanism which will automatically synthesize a group out of the result of any query returning more than one point. A group is a collection of at least two points. This allows for a workflow where common annotations can be quickly applied to large groups of points. The objective of having search and grouping, is to collect similar points, for increasing the speed of annotating metadata. Groups can be used for editing a field e.g. location for a collection of points which share the same location. We have predefined one type of group, which contains all points that relates to the same sMAP instance [4]. This is illustrated in Fig. 1, where we have a group for Building A and Building B respectively.

Each point in Metafier holds a state, which can take the value of either *Discovered*, *Validated* or *Invalidated* as defined in Holmegaard et al. [8]. This feature can minimise the effort of validating the sensor readings in other applications.

When a user is annotating metadata for a point, the last 20 readings from the point are visualized in a chart. The number of readings could easily be changed to reflect the last 24 hours

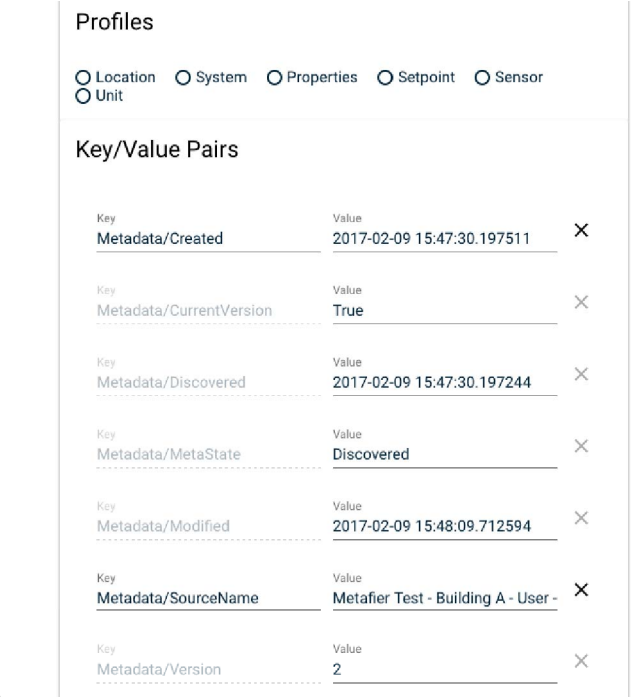

Fig. 2: Screenshot for "edit view", for a point in Metafier

or something similar. Visualized data is likely to provide the user with immediate hints towards group homogeneity and characteristics such as point type.

The purpose of a room changes over time, it could be transforming a copier room into a office. Data from points in the room will have new patterns, which could look like outliers compared to the old data. Then there is a need for updating the metadata, so that it reflects the new purpose of the room and the sensor readings can be interpreted correctly. Each time metadata have been changed via Metafier, the affected original metadata will be kept as a version. This allows the user to reason about historical data and introduces the ability to roll back to an earlier version. A calendar view – marking dates of each revision – is added to the edit view.

We have used profiles to ensure a consistent level of metadata. The building profiles provides a set of metadata keys, to have a common metadata key-set for all buildings. The idea for profiles is based on Weng et al. [22]. For the evaluation, we have created profiles for *location*, *system*, *properties*, *setpoint*, *sensor* and *unit*.

B. Implementation Details

Metafier has been implemented with Polymer Web Components [16] and uses a backend written in python. Metafier uses a key-value store like sMAP [4] for the metadata model. The points from the two building instrumentations, Building A and B, have been stored in sMAP instances. The search and grouping mechanism have been implemented as a sMAP query returning a collection of points, which is saved in Polymer with HTML5 Web Storage. Data visualization have been implemented by the Google Charts component from Polymer. Profiles are implemented as JSON schemas, which are readable and easy to configure.

III. Evaluation

A field study at the facility managers office, see section III-A been conducted. Furthermore, with a goal of having metadata for all points in buildings, a preliminary study has been conducted. We have interviewed three subjects, while they were using Metafier.

The evaluation covers the tool Metafier and not the underlying metadata model. We have introduced the features of Metafier to the test subjects shortly before they were asked to perform the tasks listed in Table I.

TABLE I: Tasks for subjects evaluating Metafier.

Task	**Description**
Login	Type user and password
Annotate a point	Enter metadata related to selected point
Annotate a group	Enter metadata related to selected points
Validate a point	Select point and click *Validate*
Create a group	Search or Click on multiple points
Use profiles	Select a profile matching selected point(s)
Save annotation	Click on save

A. Tasks related to BAS at the Facility Managers Office

For this preliminary study, we have collaborated with the facility managers at University of Southern Denmark. The facility managers maintain the building automation systems (BAS), and thereby know how the systems and subsystems interact. The facility managers have been our entry point for understanding the BAS used on University of Southern Denmark. The facility managers holds all building instrumentation information necessary to annotate the building with enough metadata to provide a metadata model rich enough to support most types of portable applications.

Based on our collaboration with the facility managers, we have analyzed their tasks related to BAS and metadata. And identified three main tasks. The three tasks are covered in section III-A1 to III-A3:

1) Control: The task of *Control*, relates to global set points and control loops influencing multiple rooms. An example could be defining the global set point for room temperature, when a room is booked. The corresponding annotation task would be to use Metafier to assign a type to a point.

2) Logic: The task of *Logic*, relates to actions or state machines within the building automation. The facility managers define actions for e.g. when a room is booked, then the light level and comfort set point should be activated. The facility manager is also linking points, such the shading height, the required lux level and the output of the lightning can be controlled by one set point. The corresponding annotation task would be to use Metafier to assign the type of control and effects of the logic to a point.

3) Inspection and Commissioning: The task of *Commissioning*, relates to test of the interconnection between the subsystems within the building. The facility manager can test whether the light level reaches the correct level, when the room is set to booked. The task of *Inspection*, relates to validate that the sensors are reporting the expected values. The corresponding annotation task would be to use Metafier to assign a boolean accepted status to a point. This relates to the process described in [8]. The task of validating points is very important for applications using data from the building, like OccuRe [11] or a personalized lighting control [13].

B. Subjects

We have for our evaluation used three test subjects with different background and knowledge about BAS and building instrumentation metadata.

1) Subject A: is a facility manager at University of Southern Denmark, with more than 10 years of experience within the field. Subject A has a background as electrical technician, and can be seen as an expert within BAS.

2) Subject B: is a student in software engineering at University of Southern Denmark, and has worked with sensor data from buildings for the last year. Subject B can be seen as a novice within BAS, but have some knowledge about metadata regarding building instrumentation.

3) Subject C: is a student in software engineering at University of Southern Denmark. Subject C can be seen as a novice within BAS, but have had an introduction to building instrumentation and software for building instrumentation.

C. Buildings and Data

For the evaluation we have used a subset of points from two different buildings. The subset has been chosen to cover one room in Building A, and two rooms in Building B. The reason for having different number of rooms, was the fact that the Building A have more than double the amount of points in the selected room. The building instrumentation is stored in sMAP instances.

1) Building A: is the OU44 building (8300 m^2 split across 4 floors) from 2015. It is located at the University of Southern Denmark, Campus Odense (Fig. 3) and contains auditoriums, teaching rooms and a few offices. The building is equipped with two building automation systems (BAS) controlling ventilation, heating and lighting. Schneider Electric StruxureWare is mainly controlling the ventilation and heating. ETS is controlling every sensor and actuation at a room level, via KNX[12].

2) Building B: is the Green Tech House (3000 m^2 split across 3 floors) from 2014. It is located at the Green Tech Center, Vejle (Fig. 4) and contains offices, canteen and a few meeting rooms. The building is equipped with one BAS controlling ventilation, heating and lighting. The BAS is a Honeywell Niagara AX, and on room level the communication is Modbus and BACnet.

Fig. 3: Building A

Fig. 4: Building B

IV. Results from Evaluation

For our evaluation described in section III, we have collected results of our subjects using Metafier. Our three subjects from section III-B have performed the tasks from Table I, while we observed the interactions in Metafier. We use the term event for either a click or a pressed key by the users of Metafier. We have recorded a timestamp for each event, which can provide the total time, from first event to last event. Annotated Points, represent the number of points from Building A and B, where the subject have either set a profile, typed a key-value pair, or annotated metadata in other ways. The maximum number of annotated points a subject could reach was 112. The results consists of a screen and audio record. We have, furthermore, have we collected all Metafier events. A summary of the collected events is shown in Table II.

From Table II we can observe that in total 2 out of 3 events were a click. Subject A have used 116 events for annotating metadata of 1 point. Subject B had a click to key ratio, where 1 out of 3 events was a click. Subject B used in average 2 events for each point. Subject C have used the mouse for 4 out of 5 events and have used 7 events per point.

TABLE II: Results from evaluation of Metafier. Results collected by JavaScript in Metafier.

Subject	Events	Clicks	Entered Keys	Total Time $[s]$	Avg. Time Between Events$[s]$	Max Time Between Events$[s]$	Annotated Points
A	116	80	36	1171	11	300	1
B	194	60	134	560	2	155	112
C	547	436	111	1193	4	87	79

A. Individual Results

We have collected data for clicks and pressed keys, the events are illustrated the timeline for each of the subjects in Fig. 5 to 7. For Fig. 5 to 7 have been used triangle markers, which have been colored based on time since last event, where darker colors mean larger delta.

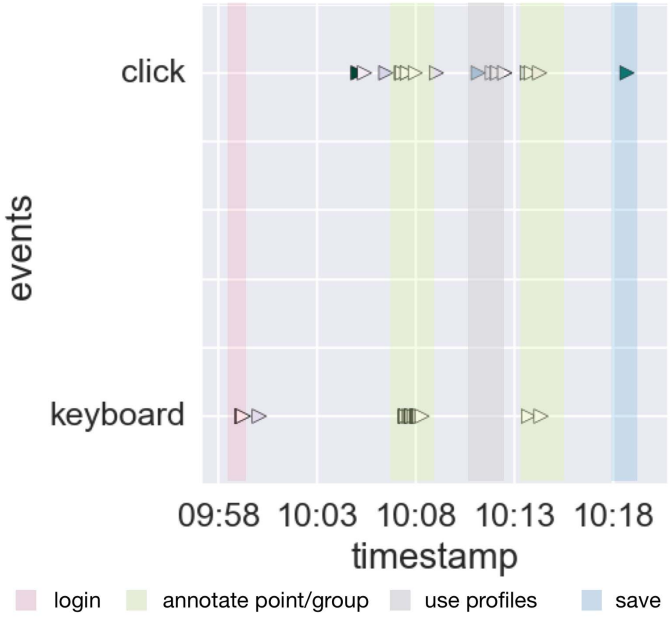

Fig. 5: Click statistics for subject A

Subject A logged into Metafier, which is shown around 09:58 in Fig. 5, then subject A had some questions regarding the tool and what the task was. At 10:08 subject A started editing a point, and then had some questions regarding that specific point. The question resulted in subject A changing to his own computer, to investigate and answer the questions regarding the point. Afterwards subject A changed values of what he had entered. A period of around 4 minutes without events occurred, from 10:14 to 10:18, due to subject A was again investigating the point on his own computer. Subject A ended by clicking on the save button. Subject A was annotating one point, where he entered information about the sensor type and location of the sensor.

Subject B logged into Metafier, which is shown around 09:55 in Fig. 6, then subject B was clicking around the tool. After 5 minutes with questions regarding the tool, subject B was editing a group of points for Building B. Subject B had some questions before editing Building A, which can be seen from 10:00, with around 2 minutes without any events. Subject B was annotating all points in this test, he entered building for

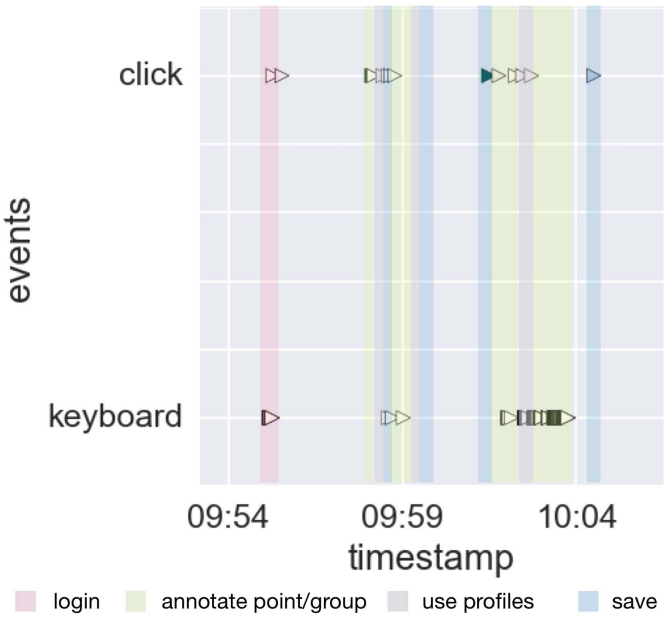

Fig. 6: Click statistics for subject B

all of them, furthermore he was setting type information for one group of temperature sensors.

Subject C logged into Metafier, which is shown around 09:13 in Fig. 7, subject C was clicking around the tool for almost 7 minutes, until 09:20. Subject C started editing Building B via the group feature. At 09:27 subject B edited 1 point from Building A and clicked on the save button. Subject C was annotating a total of 79 points, he entered location for all of those points, and for 1 point he entered type information.

V. Discussion

Based on the results in section IV, we look at how the subjects performed the tasks from Table I and the overall usage of Metafier.

We explained that a point is said to be validated when a human has verified that the data stream of the point seem to match its annotated metadata, to Subject B, whom said: *"...else ten persons working with data and every single one need to have control that it is correct what is received..."*. For a building with a high level of building instrumentation, validated and invalidated points, can be used in the commissioning phase. Subject B, whom have worked with sensor data for a year, identified and appreciated the feature of validating points.

From Table II it is clear, based on the number of annotated points and number of events, that the grouping feature was used for subject B and C. Subject B was on average using

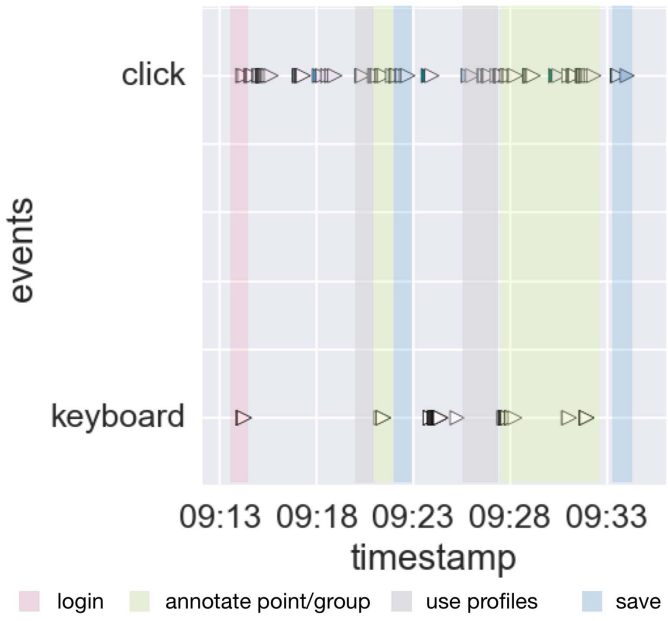

Fig. 7: Click statistics for subject C

only 2 events to process a point. Subject C said *"... as I can see it, the grouping mechanism is strong ..."* and Subject B *"... it was exactly the feature I was thinking of..."* while using the feature. By using the grouping feature, subject B could easily annotate all points in this setup. Subject B was the one using shortest time and events, but still annotating the most points.

All subjects have used a minimum of one profile, for annotating metadata. While editing a set of points, subject B said: *"... and then I just need to find a way to write celcius, which should be consistent... when I have wrote celcius once, I could chose it again..."*. Profiles in Metafier gives structure, but it could give even more structure. If Metafier was reusing earlier entered values, then values would be more consistent.

Performing our evaluation, we found that views could have had more explanatory texts. Subject A said: *"... if you should address this to me, then I would like that the descriptions was changed a bit... maybe just by hovering and then highlight with a more descriptive text..."*, and was confused about the used keys, and had to ask what he should write as values. Subject A and C needed more guidance, due to a misunderstanding of the used metadata model. Subject C needed further explanation for a couple of the keys. Furthermore we experienced a missing understanding of how Metafier should be linked with respect to tasks for facility managers and users of a building. It was like the tool was not enough, there was missing an understanding of the importance of metadata for the building instrumentation.

Subject B which had knowledge to the underlying data model, had an advantage and was the only to annotate all points for Building A and B. From this preliminary study, we have found that the tool is not enough, it is also the mindset of the users. It is important that the user of a tool to annotate metadata, has an idea of why and how the metadata should be used afterwards. In order to take advantage of the domain knowledge held by facility managers we need to convince them that metadata will benefit their work.

VI. Existing Solutions

This section compare features in existing BMS and BAS software to the features of Metafier. The existing solutions chosen, are the most commonly used ones in Denmark, which are installed in Building A and B: ETS, Honeywell Niagara AX, Siemens Desigo and Schneider StruxureWare. Based on section II-A have the same six features been chosen. The comparison is shown in Tab. III, and are based on [6, 7, 14, 19].

ETS is used for managing KNX products and it is an application for BAS. The main purpose of ETS is the ability of creating the logic between points. It has features for searching, grouping and visualization but requires a plugin. ETS does not have a feature for validation data or metadata.

Niagara AX is a full BMS which has support for search, grouping and visualization. For metadata validation, Niagara AX have a integrated type system for setting the type of a point, e.g. can a temperature sensor be linked with either $^{\circ}C$ or $^{\circ}F$ as unit.

Desigo which are the European version of Siemens BMSs, and it supports search, grouping and visualization. Desigo does have a type system like Niagara AX, where the most common types are included.

StruxureWare is a full BMS which has support for searching and grouping. For visualization it is required to create trend-logs for the points which should be visualized. A trendlog can be seen as a data storage for a specific point. StruxureWare supports data validation with a state for whether the data is validated or not. StruxureWare does have a type system like Niagara AX, where the most common types are included.

The three BMSs supports validation of metadata with respect to the most common type of points. None of these systems support validations towards a predefined metadata schema. Instead, they define all fields as optional. While this provides full flexibility it also represents a lack of formal standardization and thus a risk of deviations.

VII. Guidelines for Tool to Handle Building Metadata

We have talked with the facility managers, and they state that they will only benefit from a visualizations with sensor readings the first time they inspect a point. Validation of points relates to the same task, first time a facility manager inspect a point, they can give the point a status of validated.

While requiring an initial amount of metadata a search and group feature will increase the efficiency of metadata annotation. We have used the query language from sMAP, which can be difficult to use without knowledge of the metadata model, therefore we suggest using a simpler query language, e.g. a plain text search.

Feature	ETS[7]	Niagara AX[14]	Desigo[6]	StruxureWare[19]	Metafier
Search	✗	✗	✗	✗	✗
Grouping	✗	✗	✗	✗	✗
Visualization	✗*	✗	✗	✗	✗
Data Validation				✗	✗
Versioning					✗
Metadata Validation		✗*	✗*	✗*	✗

TABLE III: Features for annotation of metadata in existing BAS and BMS. * have been used when the feature are partial meet, or require a plugin.

The interface for facility managers, need to be simple. The used metadata model should be hidden for the end user or – alternatively – the user could be presented by a choice of view. There should be simple labels with a help text introducing the end user to what the field should contain.

The end user should be able to reuse text already entered to the system, so that spelling and typing mistakes will be minimized, or at least consistent.

VIII. Related Work

Weng et al. [22] have with BuildingDepot 2.0 created an infrastructure for building applications. BuildingDepot consists of three central services: DataService, CentralService and AppService. DataService provides the service for communicating with the physical environment, CentralService handles permissions and users, and AppService is the location of building applications. BuildingDepot has created sensor and building templates, to ensure a common level of metadata and communication within the system and to the applications using the system. The template covers the relationships between location, sensor and actuator as well as units and types of the point. BuildingDepot uses JSON via a REST API for communication. The metadata is validated against the templates. BuildingDepot does not support or have a tool for maintenance of metadata.

Balaji et al. [1] have created the framework Zodiac, which successfully classify sensors with an average accuracy of 98%. Balaji et al. [1] have used BMS tags to extract metadata. The system generates metadata automatically, but has no manual interface for maintaining metadata. They identify the problem of having mistakes in tags. Furthermore they identify the problem of having multiple tags meaning the same or slightly the same. They have used hierarchical clustering in combination with an approach similar to Bhattacharya et al. [3] to extract metadata from the tags. For solving the problem of having mistakes in tags, have been used clustering of time series data based on four groups of features (Shape, Pattern, Scale, and Texture). To obtain an average accuracy of 98%, they have combined the two approaches, tags, and data streams. The results of the two approaches were 94% and 63% for tags and data streams respectively.

Balaji et al. [2] have created Brick, which extends the key-value store to a triplet store, which is used to create a graph with relations between systems, points and all elements in the building. Brick uses RDF [20] files for storing the relationship graph and SparQL [21] for querying it. The format is open, which mean you would have a common format for all buildings. Brick does not have validation of the created model, which means you could have incorrect relations. Brick can not store historical information, like a replaced sensor, here you would have to store a new RDF file for each version of the metadata model.

IX. Conclusion

We have created Metafier, a tool for annotating and structuring building metadata. We found that search and grouping of points increased the efficiency of annotating metadata. Our preliminary study showed a simple model for validating points, which was understood by the test subjects. Overall we met challenges regarding the mindset of the end users, where the metadata model was difficult to understand. The purpose of annotating metadata was unclear for most of the test subjects. Which could be solved by an application using the metadata to solve some of the facility managers tasks. The facility manager requested for more guidance, while annotating metadata, to ensure that metadata was correctly applied. We have in section VII given guidelines for how to create a tool which can handle building metadata. The guideline does not take into account which kind of metadata model the tool uses.

X. Future Work

For this work we would like to include people with a background of user experience (UX), so that Metafier would be easier to use for the end users. This should take into account the guidelines we suggest in section VII. Furthermore we have to evaluate the tool for facility managers within other buildings than the campus of University of Southern Denmark.

Acknowledgment

We would like to thank the students and the facility managers for evaluating Metafier. This work is supported by the Innovation Fund Denmark for the project COORDICY (4106-00003B).

References

[1] Bharathan Balaji et al. “Zodiac: Organizing Large Deployment of Sensors to Create Reusable Applications for Buildings”. In: *Proceedings of the 2nd ACM International Conference on Systems for Energy-Efficient Built Environments*. ACM. 2015, pp. 13–22.

[2] Bharathan Balaji et al. “Brick: Towards a unified metadata schema for buildings”. In: *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*. ACM. 2016, pp. 41–50.

[3] Arka A Bhattacharya et al. “Automated metadata construction to support portable building applications”. In: *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*. ACM. 2015, pp. 3–12.

[4] Stephen Dawson-Haggerty et al. “sMAP: a simple measurement and actuation profile for physical information”. In: *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems* (2010), pp. 197–210.

[5] Stephen Dawson-Haggerty et al. “BOSS: building operating system services”. In: *Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation*. USENIX Association. 2013, pp. 443–458.

[6] *Desigo Insight - Operating the management station, V6.0 SP, User’s guide, Volume 1*. [Online; Accessed 2017-05-17]. 2017. URL: https://goo.gl/jVcCIM.

[7] *ETS5 Features*. [Online; Accessed 2017-05-17]. 2017. URL: https://www.knx.org/knx-en/software/ets/features/index.php.

[8] Emil Holmegaard et al. “Towards a metadata discovery, maintenance and validation process to support applications that improve the energy performance of buildings”. In: *2016 IEEE International Conference on Pervasive Computing and Communication Workshops*. IEEE. 2016, pp. 1–6.

[9] Bo Nørregaard Jørgensen et al. “Challenge: Advancing Energy Informatics to Enable Assessable Improvements of Energy Performance in Buildings”. In: *Proceedings of the 2015 ACM Sixth International Conference on Future Energy Systems*. Bangalore, India: ACM, 2015, pp. 77–82. ISBN: 978-1-4503-3609-3.

[10] Randy H Katz et al. “An information-centric energy infrastructure: The berkeley view”. In: *Sustainable Computing: Informatics and Systems* 1.1 (2011), pp. 7–22.

[11] Mikkel Baun Kjærgaard et al. “OccuRE: an Occupancy REasoning Platform for Occupancy-driven Applications”. In: *Proceedings of the 19th International Acm Sigsoft Symposium on Component-based Software Engineering*. Association for Computing Machinery. 2016, pp. 39–48.

[12] KNX Association. *About ETS*. [Online; Accessed 2017-04-27]. 2017. URL: https://www.knx.org/za/software/ets/about/index.php.

[13] Andrew Krioukov et al. “A living laboratory study in personalized automated lighting controls”. In: *Proceedings of the 3rd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*. ACM. 2011, pp. 1–6.

[14] *NiagaraAX Framework - GETTING STARTED*. [Online; Accessed 2017-05-17]. 2017. URL: https://goo.gl/j4RHoS.

[15] Christopher Palmer et al. “Mortar. io: Open Source Building Automation System”. In: *Proceedings of the 1st ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*. ACM. 2014, pp. 204–205.

[16] *Polymer Project*. [Online; Accessed 2016-01-16]. 2016. URL: https://www.polymer-project.org/1.0/.

[17] *Project Haystack*. [Online; Accessed 2015-11-17]. 2015. URL: http://project-haystack.org/.

[18] Anthony Rowe et al. “Sensor andrew: Large-scale campus-wide sensing and actuation”. In: *IBM Journal of Research and Development* 55.1.2 (2011), pp. 6–1.

[19] *SmartStruxure Solution Overview*. [Online; Accessed 2017-05-17]. 2017. URL: http://www.acscompanies.com/download/attachment/11459.

[20] W3C. *Resource Description Framework*. [Online; Accessed 2017-02-17]. 2017. URL: https://www.w3.org/RDF/.

[21] W3C. *SPARQL Query Language*. [Online; Accessed 2017-02-17]. 2017. URL: https://www.w3.org/TR/rdf-sparql-query/.

[22] Thomas Weng et al. “BuildingDepot 2.0: An Integrated Management System for Building Analysis and Control”. In: *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*. 2013.