A Pipeline for Crowdsourced IoT Data-Modeling with AI-Supported Convergence

Marc-Oliver Pahl†, Florian Bauer*, Christian Lübben*
†*IMT Atlantique, *Technical University of Munich*
† marc-oliver.pahl@imt-atlantique.fr; * {florian13.bauer,christian.luebben}@tum.de

***Abstract*—A central challenge of today's Internet of Things (IoT) is data-interoperability. Data representations vary heavily between scenarios, domains, and vendors. Interoperability requires common standards in data representation. The IoT changes fast but successful standardization typically takes time, and is often domain-specific. This paper presents a crowdsourced IoT data modeling process. Major features of the presented processing pipeline are a public open data model repository with model validation and a data-sensitive AI-based editor support that facilitates the modeling process and fosters model convergence. A conversion of 1200 data models from the biggest available IoT data model set confirms the applicability of the approach. A user study confirms the targeted enhanced usability and high data model convergence of the approach.**

***Index Terms*—IoT, data, data, modeling, interoperability, validation, ai, editor, human-in-the-loop, convergence, crowdsourcing, collaboration, open**

I. Introduction

The focus of this paper is on interoperability of IoT services via standardized data models. A central design principle of the Internet of Things (IoT) is connecting everything to create new functionality [1]. The IoT consists of distributed compute nodes that exchange data.

Data is structured by so-called data models [2]. If data is represented in a compatible way, IoT services can interact. Consequently, a central challenge of the IoT is data-interoperability. [3], [4].

A typical process for unifying data models is *standardization*. This process involves discussions of stakeholders that ideally lead to a common standard [5]. Other possibilities for reaching interoperability include patents [6] or de-facto standards e.g. via market-domination [7].

However, establishing a single standard is difficult. This reflects in existing efforts such as the project Haystack [8] where groups create data models for limited use cases (section IV).

To overcome silos, a common data format for the IoT has to be expressive. The most expressive data formats are so-called ontologies. Ontologies are typically represented as triples: $\langle \text{subject}, \text{predicate}, \text{object} \rangle$. An example is $\langle \text{wheel}, \text{belongs to}, \text{car} \rangle$.

Typical ontology representatives are the Web Ontology Language (OWL) [9] and the Resource Description Framework (RDF) [10]. Ontologies are typically not directly used as data models since they are too complex [11]. While the triples can express all kinds of information, inferring information from them is complex and thereby time-intense [12], [13]. Consequently, ontologies are too general, slow, and have a too bit-intense representation for the specific and highly-optimized real-world IoT data exchanges.

With the Virtual State Layer (VSL), a data-centric IoT middleware exists that uses a lightweight ontology as data model [14]. It provides the expressiveness of an ontology with the usability of a Domain Specific Language via its modular design [15]. The VSL data modeling follows a collaborative, crowdsourcing [6] approach [4]: the VSL IoT data model repository is public and open for contributions.

Standardization of data models happens via publicly available usage-statistic-based charts. This approach is scalable and suitable for the IoT with its numerous heterogeneous devices, applications, and use cases. Despite having advantages, the VSL modeling approach did not get widely adopted. A reason for that might be that it has two major shortcomings that can be summarized in the following questions:

- How to ensure model-integrity in a fully-decentralized ontology-creation process?
- How to foster convergence of data models in the wide IoT domain with its numerous data models when enabling distributed contributions?

Answering the identified shortcomings of the crowdsourced data modeling approach, this paper introduces an editor-based AI-supported pipeline that solves both open issues. To ensure the integrity of data models checked into the open repository, it introduces automated validation with rejection of faulty data models. To ease the data model creation process, and to foster convergence, a data-sensitive editor is introduced that proposes auto-completions of data models based on a machine-learning-supported classification of models.

A conversion of more than 1200 data models from the biggest identified open IoT data model repository confirms the applicability of the approach. A user study confirms the usability of the approach.

Section II introduces relevant related work in the areas standardization and ontology editors. VSL ontology that proved to be most suitable for our goal. Section III introduces the proposed pipeline for crowdsourced IoT data-modeling with AI-supported convergence. Section IV evaluates the pilot quantitatively regarding the performance of the used AI approach and qualitatively with a user study.

Funded by the German BMWI in DECENT (0350024A) and the Brittany region via FEDER funds of the EU (Cyber CNI).

978-3-903176-32-4

II. Related Work

The related work of this paper can be structured into the parts *standardization*, and *editor-support*.

A. Data Model Standardizations

Existing IoT standardization mechanisms rely on expert intervention. This is not suitable for the intended scaling and collaborative editing process.

The Web Ontology Language (OWL) by the World Wide Web Consortium (W3C) does not implement global convergence mechanisms. OWL ontologies are manually converged. A manual collaborative process is proposed in [16].

Project Haystack is an open-source initiative for standardizing IoT data models. Haystack introduces semantic tags for identifying functionality in data models [8]. Haystack's tags cover the semantic meaning and behavior of the tagged models. With version 4.0, ontology modeling is supported. It enables developers not only to use predefined tags and schemas but also to customize them. This enables a broader use. Still, no global convergence mechanism exists or could be easily implemented. Furthermore, object-orientated inheritance which fosters reuse and semantic convergence is missing.

IotSchema [17] is an extension of schema.org for the IoT. It is developed in an open community process using a W3C community group and Github. IotSchema covers a large number of property values, things, tags, capabilities, and interaction patterns. Although IotSchema is developed within a community process, there is no build-in, automatic standardization process. Repository maintainers have to converge proposed models explicitly.

BrickSchema [18] is a uniform metadata schema for buildings. It includes classes and their relationships as well as tags for different entities. Schemas are stored in a public Github repository. There exists no automatic standardization pipeline.

The closest related work regarding standardization is [4]. The authors propose an automated evaluation of the use of data models. The resulting usage-frequency charts can be used by human modelers to converge to common data models.

This paper extends their approach by automatic model validation, automated tagging, and actively supporting data modelers with suggestions in the data model creation process.

B. Ontology Editors

Different editors for ontology data models exist. Several editors support collaborative work. None encourages developers to reuse existing data models. Though providing rich support features, they miss active support in the model creation.

The most widespread ontology editors are Protégé, OntoStudio, Swoop, TopBraid Composer, and CONGAS. Protégé is a free open-source editor. Its online version is WebProtégé [19]. The OntoStudio ontology editor (OntoEdit) uses client/server architecture based on the Eclipse Framework [20]. Swoop (Semantic Web Ontology Overview and Perusal) is an ontology editing, browsing and visualization tool [20]. TopBraid Composer also uses the Eclipse platform, allowing the modeling of ontologies with OWL and RDF. It supports reasoning and consistency checking methods [21]. The Collaborative Ontology development framework based on Named Graphs (CONGAS) is built on top of Semantic Turkey, a Knowledge Management and Acquisition System. Its key feature is collaborative ontology development [22].

TABLE I
Comparison of crowdsourced ontology development editors.

Editor Name	User-friendly IDE Features	Syntax Validation	Collaboration Support	Convergence Mechanisms
Protégé	+	+	+	−
Apollo	+	+	−	−
OntoStudio	+	+	+	−
Swoop	+	+	−	−
TopBraid-Composer	+	+	+	−
CONGAS	o	−	+	o

The following assessment focuses on syntax validation, global data-model-convergence mechanisms, collaboration-support, and usability via providing an Integrated Development Environment (IDE). Table I summarizes the comparison results. The table shows if an editor completely fulfills a criterion (+), partially fulfills it (*o*), or completely misses a point (−). Every editor behaves similar-to, or actually is an IDE with default features including a text editor with syntax highlight and similar features (+). The only exception is CONGAS, which is a Knowledge Management and Acquisition System. Still, it includes some features typical for an IDE (*o*). Comparing the editors on syntax validation shows that all IDE-like editors support syntax validation out of the box (+). Only CONGAS entirely misses this feature (−). Apollo and Swoop do not support collaboration (−). The other editors do (+). Crowdsourced convergence mechanisms are only available in CONGAS, but it does not foster the reuse of existing models. It is possible to converge data models by reusing branches of other developers. However, developers are not encouraged to do so (o). Other editors completely miss this feature (−).

Summarizing, most editors offer convenience features, plug-in support, syntax highlighting, and collaboration support. However, no editor implements methods to develop data models in a crowdsourced way including suggestions to reuse suitable, existing models for fostering interoperability and convergence.

III. Data Modeling Pipeline

The proposed pipeline for crowdsourced IoT data-modeling with AI-supported convergence consists of four parts: Model validation (III-A), AI-supported tagging (III-B), AI-supported Auto-Completion (III-C), and Front-end (III-D).

Figure 1 illustrates the system architecture for the pipeline. It is structured in three parts. The left side is the front-end running on a data modeler's computer. It provides the Eclipse-based front-end that includes XSD-based syntax checking. The middle layer can run locally on the left, or in the cloud on the right. It does the tag-proposal and model-comparison.. The right part finally runs in the cloud. It does the semantic

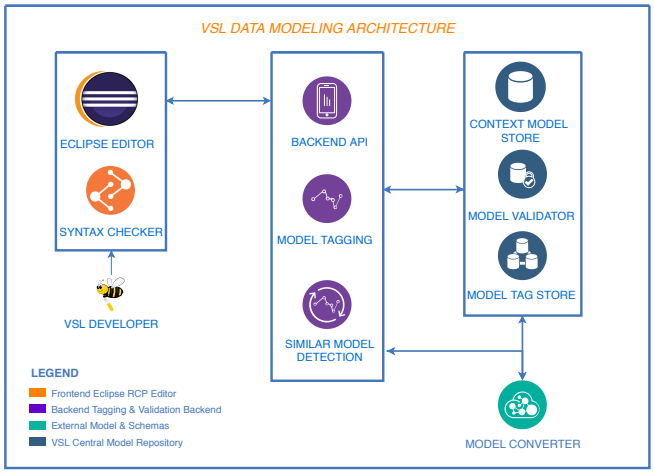

Fig. 1. Proposed Pipeline for Crowdsourced IoT Data-Modeling with AI-Supported Convergence.

validation, stores and provides all valid VSL data models, and provides a store for semantic tags.

A. Model Validation

The VSL ontology bases on four basic data types [4]:

- TEXT, for representing textual values
- NUMBER, for representing numeric values
- LIST, for dynamically generating data at run-time
- NONE, for nodes that do not carry values

More basic types could be defined. However, these types proved to be sufficient to express IoT data.

VSL data models are defined over a global GIT-based *Central Model Repository* (CMR). Consequently, all valid data model identifiers can be found there. It is important that only valid VSL data models can be checked into the CMR, as invalid data is a major reason for malfunctioning IoT components. Having the validity always checked increases the resilience of an IoT system significantly.

The previous explanations motivate the following validation algorithm. It has four steps: 1. Resolve all dependencies, 2. Check all restrictions, 3. Minimize the new data model, and 4. Store the minimized model.

The implementation of the validator happens via a GIT post-commit hook in the GIT server as shown at the right of the processing pipeline diagram in fig. 1.

B. AI-supported tagging

Tags can be used to define the semantics of a value in a VSL data model. For fostering reusability, it is important that users tag their VSL data models correctly. A fridge that is called bathtub is unlikely to be reused. Tagging requires expertise since tag categories have to be known or newly defined.

Following the VSL approach [4], tagging can be done in a crowdsourced way. The usage frequency of a tag can be taken as a measure for its relevance. While humans can learn the tagging process from a big and diverse enough data set, fuzzy machine-learning algorithms can do that too. The advantage of using a machine-learning algorithm is that it is inter-subjective. In addition, ML can reduce the training time for humans, making the modeling process better accessible. This is important as crowdsourced modeling targets regular users to do modeling when they add new so-far-nowhere-supported functionality to their IoT systems, which implies a rare use.

Based on the existing data models in the CMR, a tag space is created that consists of the model identifiers. The probability that the right tags are chosen increases by pre-filtering the search space. The later in the VSL inheritance chain, the more general is a data type, e.g. “specialFan,fan,airblowingDevice” [4]. This suitably reflects in their frequency, which motivates that they will be more often proposed as tags. This is helpful for the auto-completion or proposition of data models (section III-C).

While entering a data model, the tagger continuously creates tag proposals and offers them to the the pipeline for further usage in section III-C. In addition, tags are proposed when saving a VSL data model.

For identifying structural similarities, it comes in handy that VSL data models are stored in a minimized form (section III-A). Using minimized models, similarities can be identified more easily as only differences are stored in each VSL data model, enabling a step-wise comparison.

The pilot implementation uses supervised learning to associate semantic information with tags. Lists of available data models and associated tags are used as training data set. In the prediction phase, the predictor takes user-edited data models as input. After a semantic analysis, the list of best-fitting tags and their associated models is provided. These associated models are the existing data models that should be reused or extended. After adding a new data model, the predictor is retrained with the extended list of data models.

Multiple approaches are suitable. The search considers two shortcomings of the dataset: imbalance of the labels, and a low number of samples (1230) in relation the number of labels (88). Random Forests (RF), AdaBoost and XGboost classifiers are traditional approaches. All belong to the family of ensemble-methods. They either rely on bootstrap aggregation (bagging;RF), or boosting (AdaBoost, XGBoost) techniques. Both techniques train multiple weak learners and combine them into one classification. Bagging trains learners in parallel while boosting connects learners iteratively to create a strong learner.

Another possibility are Neural Networks. Among this class, Recurrent Neural Networks (RNN) are very-well suitable. Through internal state preservation they are able to identify semantic information [23]. To evaluate their performance in extracting semantic information from data models and their ability in handling the weaknesses of the dataset, the predictor of the evaluated pilot bases on a RNN.

Within the evaluation (Section IV), the resulting RNN model takes a vector of 18 features as input and has 155 possible labels. The hidden layers consist of 2 LSTM layers

with 128 neurons each. Dropout is used for regularization to mitigate over-fitting.

C. AI-supported Auto-Completion

The described tagging algorithm likely returns suitable VSL types for a newly created data model. The reason is that inherited types –that are more on the right of the inheritance chain and therefore describe more general concepts– are statistically more-frequent and therefore more-likely proposed. This is helpful for the auto-completion of data models.

The editor proposes more-basic VSL data types to the user. By simply clicking on a proposal, the inherited model is loaded and all its data is added in the editor window. The automatically added information from the inherited type helps by showing all already defined fields. It is later stripped by the validation process (section III-A). Having partially-pre-filled data models can speed-up the modeling process, and increase the quality especially for less-experienced users. It also makes reuse more likely, which is desired [4].

D. Editor Front-end

For the front-end, functionality such as an input window, a possibility for user interaction, and custom validation logic are required. The Eclipse editor [24] offers such functionality. Consequently, a plug-in for it is developed for the prototype.

The VSL data models are represented in XML. This allows creating an XSD for validating their syntax. The Eclipse editor framework offers built-in support for this kind of validation as shown on the left of fig. 1.

The model-tagging and similarity-detection are implemented as plug-ins. The model-tagger and similarity-checker in the middle of fig. 1 continuously exchanges with the CMR.

IV. Evaluation

This section covers an evaluation of the ML predictor within the tagging back-end and a user study that evaluates the benefit using the model editor regarding usability and model reuse.

A. ML evaluation

Evaluating the RNN approach requires a data set. The evaluation makes use of the biggest public IoT data model sets identified: Project Haystack, IoTSchema, and BrickSchema. Their repositories contain data models with semantic tags.

An implemented model converter produces tagged VSL context models from all three data model repositories. The conversion process shows that the VSL data model can express the semantic meaning of the other models. Converting all models results in around 1200 tagged data models in the VSL context model description format. These are used to train the tagging back-end.

The predictor is based on an RNN as proposed in III-B. In the evaluation phase, it was trained with 80% of the converted data models, and validated with the remaining 20%. The results show that the RNN tends to predict the most frequent labels. Analysis of the input data reveals that the tag distribution is highly imbalanced. This comes from the fact that the models are driven by the Heating Ventilation and Air Conditioning (HVAC) community. The identified imbalance leads to a bias of the RNN towards a small number of frequent tags. In addition to this imbalance, the number of training samples is relatively small compared to the number of labels. The data set contains 1230 data models with 88 different tags. Consequently, the use of the RNN appears to be promising but requires further testing with more data when available.

B. User study

The goal of the user study was measuring user acceptance of the model editor. In addition, the time saving and the reuse of models were quantified. With regards to the typical workflow, for respecting the learned tag space, users have to implement a VSL context model of an air conditioner.

The experiment is two-fold: In the first step, users have to create the context model without AI-based support. In the second step, they use the model suggestion. The comparison of both cases enables the ability to highlight differences.

The user acceptance is reflected though questions targeting the ease-of-use and benefit of model suggestions. The time for creating the models is taken to compute the savings when applying model suggestions. The collection of created models for both cases allows the analysis of reused elements.

Even though the results are not representative yet, they show very positive user acceptance. The time for creating a model could be reduced by 25%. The reuse of models was measurably encouraged: all users were able to identify similar models for reuse. This results indicate that the goal of model convergence is fostered by the proposed approach as intended.

V. Conclusion

This paper tackled a central challenge of today's Internet of Things: data-interoperability. Using a 2014 proposed approach for crowdsourced IoT data modeling as base, this paper showed how an editor pipeline can significantly improve a collaborative convergence and user acceptance, while keeping the openness of the original proposal.

Section II showed that existing ontology editors miss the functionality proposed in this paper. In section III, an editor-centric pipeline for crowdsourced IoT data-modeling was introduced. Its central concepts are supporting the human model editor at create time with syntactic and semantic validation of the entered data model information. In addition, this paper proposed using AI-based recommenders for tagging data models, and for auto-completing them in the creation process.

The evaluation in section IV showed that the approach works. It also showed that the approach requires a bigger data set for providing useful results.

The presented solution can serve as a blueprint for adding AI-support to other data-model and data-model editors as well as to crowdsourced data modeling. We hope that it can help making more IoT devices and services interoperable in the future.

REFERENCES

[1] M.-O. Pahl, G. Carle, and G. Klinker, "Distributed Smart Space Orchestration," in *Network Operations and Management Symposium 2016 (NOMS 2016) - Dissertation Digest*, 2016.

[2] A. Pras and J. Schoenwaelder, "On the Difference between Information Models and Data Models," Internet Engineering Task Force, Tech. Rep., 2003.

[3] M.-O. Pahl, "Multi-Tenant IoT Service Management towards an IOT App Economy," in *Hot Topics in Network and Service Management (HotNSM) at International Symposium on Integrated Network Management (IM)*, Washington DC, USA, Apr. 2019.

[4] M.-O. Pahl and G. Carle, "Crowdsourced Context-Modeling as Key to Future Smart Spaces," in *Network Operations and Management Symposium 2014 (NOMS 2014)*, May 2014, pp. 1–8.

[5] A. Y. Ding, J. Korhonen, T. Savolainen, M. Kojo, J. Ott, S. Tarkoma, and J. Crowcroft, "Bridging the gap between internet standardization and networking research," *Computer Communication Review*, vol. 44, no. 1, pp. 56–62, 2013.

[6] J. L. Contreras, "A tale of two layers: Patents, Standardization, and the Internet." *Denver Law Review*, vol. 93, no. 4, 2016.

[7] A. Brem, P. A. Nylund, and G. Schuster, "Innovation and de facto standardization: The influence of dominant design on innovative performance, radical innovation, and process innovation," *Technovation*, vol. 50-51, pp. 79–88, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.technovation.2015.11.002

[8] D. N. E. S. Prairie, J. P.-H. Petze, and M. P.-H. Petock, "Project Haystack - A CABA white paper," vol. 66, pp. 37–39, 2012.

[9] W. O. W. Group *et al.*, "Owl 2 web ontology language document overview," *http://www. w3. org/TR/owl2-overview/*, 2009.

[10] R. Cyganiak, D. Wood, M. Lanthaler, G. Klyne, J. J. Carroll, and B. McBride, "Rdf 1.1 concepts and abstract syntax," *W3C recommendation*, vol. 25, no. 02, 2014.

[11] L. Obrst, C. Werner, M. Inderjeet, R. Steve, and B. Smith, "The Evaluation of Ontologies: Toward Improved Semantic Interoperability. Dans JO Christopher, Baker, & K.-H. Cheung," *Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences*, pp. 139–158, 2007. [Online]. Available: https://philpapers.org/archive/OBRTEO-6.pdf

[12] Y. B. Kang, J. Z. Pan, S. Krishnaswamy, W. Sawangphol, and Y. F. Li, "How long will it take? Accurate prediction of ontology reasoning performance," *Proceedings of the National Conference on Artificial Intelligence*, vol. 1, pp. 80–86, 2014.

[13] Y.-b. Kang, Y.-f. Li, and S. Krishnaswamy, "Using Ontology Metrics," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7649 LNCS, no. PART 1, pp. 198–214, 2012.

[14] M.-O. Pahl and S. Liebald, "Designing a Data-Centric Internet of Things: VSL," in *Int. Conference on Networked Systems (NetSys)*, 2019.

[15] ——, "A Modular Distributed IoT Service Discovery," in *Int. Symposium on Integrated Network Management (IM)*, 2019.

[16] V. Ludovici, F. Smith, and F. Taglino, "Collaborative ontology building in virtual innovation factories," in *Collaboration Technologies and Systems (CTS), 2013 International Conference on*, 2013, pp. 443–450.

[17] Schema.org. iotschema.org. [Online]. Available: http://iotschema.org

[18] B. Schema. Brick - a uniform metadata schema for buildings. [Online]. Available: https://brickschema.org/

[19] T. Tudorache, C. Nyulas, N. F. Noy, and M. A. Musen, "WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the web," *Semantic Web*, vol. 4, no. 1, pp. 89–99, 2013.

[20] S. C. Buraga, L. Cojocaru, and O. C. Nichifor, "Survey on Web Ontology Editing Tools," *Transactions on Automatic Control and Computer Science*, vol. NN, no. ZZ, pp. 1–6, 2006. [Online]. Available: http://www.aifb.uni-karlsruhe.de/WBS/ysu/publications/OntoWeb_Del_1-3.pdf

[21] E. Alatrish, "Comparison Some of Ontology Editors," *Journal of Management Information Systems*, vol. 8, no. 2, pp. 018–024, 2013.

[22] D. Bagni, M. Cappella, M. T. Pazienza, and A. Stellato, "CONGAS : a CO llaborative ontology development framework based on N amed G r A ph S," pp. 1–10.

[23] X. Guo, J. Ma, and X. Li, *LSTM-Based Neural Network Model for Semantic Search*, 01 2020, pp. 17–25.

[24] "Enabling open innovation & collaboration — the eclipse foundation," https://www.eclipse.org/, (Accessed on 10/20/2020).