Semantic Modelling of Plans and Execution Traces for Enhancing Transparency of IoT Systems

*Milan Markovic, †Daniel Garijo, *Peter Edwards and *Wamberto Vasconcelos
*University of Aberdeen, Computing Science, Aberdeen, UK.
Email: {milan.markovic,w.w.vasconcelos,p.edwards}@abdn.ac.uk
†University of Southern California, Los Angeles, CA, USA.
Email: dgarijo@isi.edu

***Abstract*—Transparency of IoT systems is an essential requirement for enhancing user's trust towards such systems. Provenance mechanisms documenting the execution of IoT systems are often cited as an enabler of such transparency. However, provenance records often lack detailed descriptions of a system's expected behaviour. Plan specifications describe the steps needed to achieve a certain goal by a human or an automated system. Once plans reach a certain level of complexity, they are typically decomposed in different levels of abstraction. However, this decomposition makes it difficult to relate high level abstract plans to their granular execution traces. This paper introduces EP-Plan, a vocabulary for linking the different levels of granularity of a plan with their respective provenance traces. EP-Plan also provides the means to describe plan metadata such as constraints, policies, rationales, and expected participating agents associated with a plan.**

I. Introduction

An increasing number of systems provide means to document provenance records of past executions [1] (i.e., *execution traces*), in order to inspect and explain the creation process of existing results or behaviour of a system. In this paper, we argue that recording such provenance can be further enhanced by recording the set of planned steps that guided its execution, and we refer to such records as *plans*.

Plans document intended system behaviour, which is beneficial at the point when no runtime provenance is available (e.g. to assess risks associated with a planned IoT deployment). Plans are also critical to understand errors, enabling a point of reference for comparison when the execution deviates from what was planned to happen.

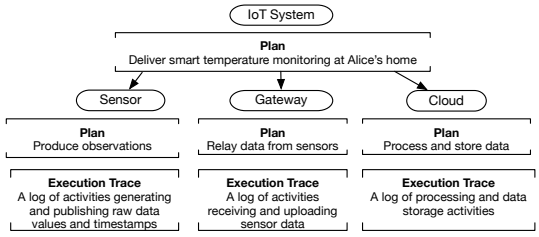

Fig. 1. An illustration of different granularity levels of plans and execution traces in an IoT system.

Plan specifications may become very complex, and therefore users tend to simplify them into smaller plans. For example, Figure 1 illustrates a simplified view of an IoT system which follows a high level plan for delivering smart home temperature monitoring. At a finer detail level, such a system consist of individual components (e.g., sensing devices, web services) that follow their specific individual plans (e.g., a temperature sensor will observe air temperature and upload readings via a gateway device to an online location). Each of these devices might generate execution traces at different levels of detail which are disconnected from the more abstract high level descriptions of the expected and the actual system behaviour.

In this paper, we argue that semantic technologies such as ontologies and linked data[1] provide a suitable method to define a machine processable model of domain concepts, such as IoT devices, and their inter-relationships, which can be shared and reused across different IoT deployments and reasoning applications. Applying such technologies to the IoT domain is not a novel idea and a number of semantic resources exist [2].

Individual execution traces can be sufficiently documented using the W3C standard on provenance (PROV) [3]. However, detailed descriptions of important aspects of a plan specification still remain a subject of discussion within the provenance community, and the link between different levels of abstraction of a plan and the corresponding execution traces remains largely unexplored. PROV acknowledges the need to associate a plan (*prov:Plan*) with the execution of an activity (*prov:Activity*), but does not specify how a plan may be further defined or decomposed. In previous work, Garijo and Gil proposed P-Plan [4], a vocabulary designed to link scientific processes to provenance traces. However, a comprehensive representation of levels of abstraction in plans and capture of additional contextual information related to the planned processes was not within the scope of P-Plan. In this paper we present a domain agnostic model called EP-Plan for describing plans and their execution traces which builds on P-Plan [4] and PROV-O [3]. Furthermore, we discuss its application in the IoT domain and present a series of example queries executed against our test knowledge base.

The following contributions are discussed in this paper:

- A set of requirements for capturing critical plan metadata to enhance transparency descriptions of IoT systems and

[1]https://www.w3.org/standards/semanticweb/data

978-1-5386-5541-2/18/$31.00 ©2018 IEEE

associating different levels of abstraction in plans with their execution traces.
- A vocabulary implementation designed to address the aforementioned requirements.
- A repository of publicly available materials including EP-Plan examples, test datasets, and a series of competency questions formalised as SPARQL queries.

The remainder of this paper is structured as follows: Section II introduces an IoT use case that motivated the design of EP-Plan, and which we used to define the ontology requirements in Section III. Section IV describes the core concepts of EP-Plan. Section V discusses example queries performed on a test dataset described using EP-Plan; Section VI discusses related ontologies and alignments. Finally, Section VII concludes the paper with discussions of future work.

II. IoT Use Case

The IoT use case was inspired by several activities in the *Trustlens* project[2], which explores various facets of what it means to realise trusted IoT ecosystems, with a particular focus on aspects of governance, transparency and accountability.

IoT deployments are often complex ecosystems consisting of multiple IoT devices and a range of supporting cloud-based components. Users impacted by IoT are usually unable to assess these systems due to their lack of transparency, which is required for supporting trust between citizens and such infrastructures [5], [6].

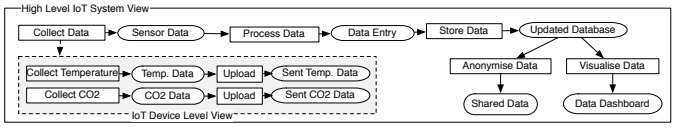

Fig. 2. An example process and data flow of an IoT monitoring system measuring temperature and CO2 levels.

The *Trustlens* project held a series of workshops and found that real users encountering IoT technologies ask many transparency questions including: *What data is being collected? Why is it being collected and who is collecting it? Who is managing the system? Is the system secure?* [7].

Such questions become even more pressing during community consultations on future IoT deployments that might be led, for example, by local councils before a public IoT deployment. These findings suggest that it has to be possible to audit IoT systems in terms of their expected and actual behaviour. However, to enable auditing, appropriate models are needed for representing important aspects of system behaviour at various levels of detail that are required by different use cases.

Fig. 2 illustrates a simplified view of processes and data flows in an example IoT deployment. At a higher abstraction level, IoT devices are used to collect sensor readings which are then processed and stored by cloud-based services. A dashboard is used by the local council to visualise data and anonymised reports are also shared with third parties. This level of abstraction is suitable for answering general queries that human users might have (e.g., *What is the system doing?*), however, it might be insufficient for other assessments which require access to detailed data provenance (e.g. data quality assessments).

At a finer grained level, the data collection process may be decomposed into a series of processes controlled by an IoT device (e.g., sensing, data transfer, etc.) that follows its own local plan. Such data collection processes can be performed by many different devices which could follow different plans (i.e. they produce the same results but contain different steps). This information would not be available at a higher level of abstraction. However, to be able to assess, for example, whether an IoT device poses potential risks to users' privacy, information about agents that control data handling processes, device policies (e.g., those ensuring the compliance with government regulations), and constraints resulting from these policies need to be made available for assessment [8].

To answer some of the questions identified by the *Trustlens* project, it is possible to utilise existing vocabularies such as the W3C Semantic Sensor Network ontology (SSN) [9], SDPO [10], PROV-O [3], and P-Plan [4]. However, models for representing the planned behaviour such as P-Plan, lack the ability to represent important concepts such as constraints, agency, and policies [10], [8]. Such models also do not capture representations of objectives that such systems intend to achieve, which are critical for answering questions related to its purpose. Similarly, to further address questions querying the *why* aspects of IoT deployments, rationales for inclusion of individual plan elements should also be recorded. For example, consider a system with a security policy that requires encryption of all internet communication. This policy provides a rationale for the inclusion of a constraint that requires the step uploading sensor data to the cloud to use a secure connection. In turn, a broader data protection legislation provides a rationale for the enforcement of such a security policy as part of the plan specification of an IoT system.

Similarly, the objectives of an IoT deployment (e.g., to monitor temperature levels in social housing for purposes of preventive maintenance) may be linked to a rationale recorded in a strategic plan decided by a local council. Capturing such rich information about plans executed by IoT systems enables them to be linked to a wider deployment context and thus enhances the transparency of such systems - something that is necessary to enhance users' trust.

III. Modelling Requirements

This section describes modelling requirements for the EP-Plan model to provide a vocabulary for capturing plans and execution traces of IoT systems. The section focuses on the novel contributions of the model and hence does not include some basic requirements for plan descriptions. These relate to the ability to represent individual planned processes, their inputs and outputs, individual elements of the execution trace (i.e. activities and entities) and their correspondence to the

[2]https://trustlens.org

plan, which are addressed by the models EP-Plan extends (PROV-O [3], P-Plan[4]) and thus are not discussed here.

1) It should be possible to link different levels of plan abstractions and their corresponding execution traces: A varying granularity of plans and corresponding provenance is a common result of distributed executions and therefore multiple abstractions of plans are needed for different types of assessments. Linking such specifications is vital for documenting a coherent account of the behaviour of a system.

2) It should be possible to capture associations between agents and the portions of plans in which they are allowed to participate: Plan executions are often associated with a number of agents executing portions of the overall system plan. Detailing individual responsibilities of agents as part of plan specifications is crucial, for example, to support risk assessment in plans that process personal data.

3) It should be possible to document references to constraints and policies that are associated with a plan: Constraints in plans commonly define restrictions such as required value range and location of inputs, types of permitted data transfer methods used, etc. Such constraints are often the result of policies. Policies [11], [12], represent concepts like permission, obligation and prohibition, and establish the activity/action which is controlled by the policy, as well as the role/party to whom the policy is directed. Policies can provide additional context for constraints, for example, by establishing what sanctions/rewards are applicable to whom, if the constraint is not met (or is only partially met). To illustrate, consider the European General Data Protection Regulation (GDPR[3]) that establishes how personal data should be handled, as in, for instance, the need for encrypting data in some contexts and where data can be stored. Note that we do not seek to represent a comprehensive vocabulary for defining constraints and policies which can be represented by existing models (see Section VI).

4) Plans should be able to document how data between individual processes should be exchanged: The information about how data is exchanged between different parts of the system is important, especially in the context of security and privacy assessments. To enable constraints to be recorded, such as those requiring an encrypted communication (see REQ3), the specification of a data exchange between processes needs to be captured in a plan specification.

5) It should be possible to capture plan objectives, their relations to other plan elements, and rationales associated with a plan: Plans may become complex data flow descriptions. Identifying the relevant final products of a plan (e.g., a variable representing the result of a calculation) is not always straightforward. Objectives provide information about the purpose of a plan (e.g., a city council wants to monitor occupancy levels in properties they manage). Objectives might be associated with specific plan elements that are expected to achieve them (e.g., data representing an estimate of occupancy levels calculated based on CO2 sensor readings). Similarly, descriptions of rationales help with understanding why some of the plan elements are present in the plan (e.g., a constraint being a result of a policy that is enforced due to GDPR).

IV. Describing IoT System Plans and Execution Traces Using EP-Plan

In this section we describe how the EP-Plan ontology can be used as a core vocabulary to provide descriptions of IoT system plans and to link the corresponding execution traces in accordance with the modelling requirements identified in Section III.

Semantic descriptions of IoT systems are recorded in the form of directed acyclic knowledge graphs. In such graphs, nodes are referred to as individuals and represent the instances of specific concept types defined in the appropriate ontology. For example, an individual representing a single process in a plan can be described with a type *ep-plan:Step* defined in the EP-Plan ontology. Ontologies also define edges that can be used to connect different nodes of the graph and are referred to as object properties. Object properties describe the links/relationships between the individuals. For example, an individual of type *ep-plan:Step* can be linked to the individual describing the overall plan (i.e. of type *ep-plan:Plan*) via relationship *ep-plan:isStepOfPlan*. If an individual is linked to literal values (e.g. text) such relationships are referred to as data and annotation properties.

A. Modelling Plans of IoT Systems

EP-Plan[4] utilises *steps* to denote any planned processes, and *variables* to represent inputs and outputs of steps. Both steps and variables belong to a *plan*. Figure 3 illustrates a partial description of a simple plan denoting the flow of data between a temperature sensing process and a data storage process.

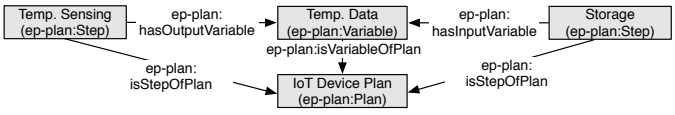

Fig. 3. An example description of a simple plan.

We will now discuss how such descriptions can be enhanced with additional metadata to further contextualise IoT deployments.

1) Plan Agency and Responsibility: In some cases, only certain agents (*ep-plan:ResponsibleAgent*) are allowed to assume a degree of responsibility over the execution of plans or individual planned steps. In EP-Plan, agents (*ep-plan:ResponsibleAgent*) can be linked to such steps using the property *ep-plan:hasPermittedAgent*. If a plan contains an element that is linked with *ep-plan:ResponsibleAgent*, that agent will also be linked to the plan specification using *ep-plan:includesResponsibleAgent*. Figure 4 includes a description of an *AWS* agent (denoting the Amazon Web Service) that is expected to perform data storage.

[3]https://eugdpr.org/

[4]ep-plan namespace: http://w3id.org/ep-plan

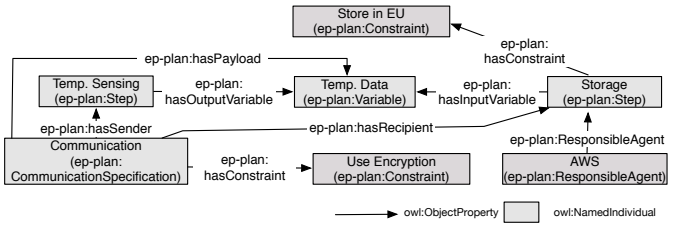

Fig. 4. An example description of a simple plan enriched with references to constraints and agents.

2) Plan Constraints: ep-plan:Constraint represents any kind of restriction associated with *ep-plan:Plan* and its elements. A constraint defines pre and post conditions restricting various attributes of the target element (e.g., time when the activity started, location of the input data, etc.), as well as the kind of entities used and generated, characteristics of agents permitted to assume responsibility for an activity, etc. Figure 4 illustrates a constraint description specifying that the storage step should be performed within the European Union (EU). Another constraint (*Use encryption*) restricts the communication of data between the sensing and storage step to use encryption. Here, *ep-plan:CommunicationSpecification* represents a planned specification of variables communicating between steps, which is then linked to steps via *ep-plan:hasSender* and *ep-plan:hasRecipient*.

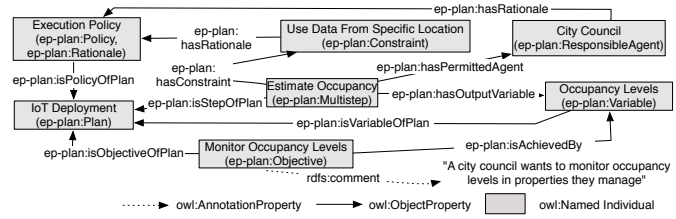

Fig. 5. An example description of a simple plan enriched with references to policies, rationales, and objectives.

3) Plan Policies, Rationales and Objectives: ep-plan:Policy describes a group of rules (i.e. permissions, obligations, and prohibitions) that can be associated with a plan, along with its associated metadata. Policies may affect a plan or individual plan elements, referring to diverse aspects such as data protection, security, plan execution, etc. Fig. 5 illustrates a policy that motivates the inclusion of a constraint restricting the data collection process to a specific location and definition of an agent (*City Council*) that is permitted to execute the *Estimate occupancy* step.

ep-plan:Rationale may be used to represent any resource (including elements of a plan) that provides some explanation about why a plan element is part of the plan. Plan elements can be linked to rationales via the *ep-plan:hasRationale* property (e.g., see the links between execution policy, city council and location constraint in Fig. 5). Similarly, *ep-plan:Objective* represents objects that provide information about the expected purpose of a plan. The relationship *ep-plan:isAchievedBy* links objectives to portions of a plan that are expected to contribute to delivering these objectives. An example is shown in the lower part of Fig. 5, where the *Occupancy levels* output variable is the main objective of the *IoT Deployment* plan.

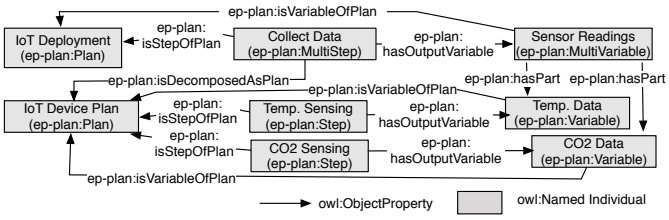

Fig. 6. An example partial description of a plan and its sub-plan.

4) Plans & Sub-plans: Plans described at finer grained level can be linked as sub-plans to their more abstract counterparts (see Figure 6). EP-Plan supports descriptions of composite steps as *ep-plan:MultiStep* and links to the corresponding sub-plan (*ep-plan:Plan*) via *ep-plan:isDecomposedAsPlan*. An *ep-plan:MultiVariable* represents an aggregation of variables in the more abstract plan that are described in more detail in a sub-plan specification. *ep-plan:hasPart* is used to link variables to their aggregate abstractions (*ep-plan:MultiVariable*) across different plans. Fig. 6 illustrates an example where a step for data collection from a high level system plan is decomposed as a more detailed sub-plan executed by a specific IoT device.

B. Linking Execution Traces to Plans

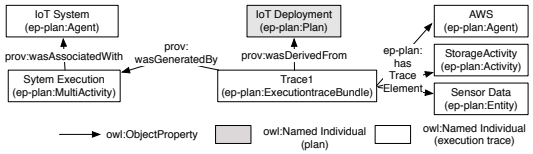

Fig. 7. An example partial description of a plan and mechanism for describing individual bundles of execution traces.

A single execution trace is contained by *ep-plan:ExecutionTracebundle* (this is also a type of *ep-plan:Entity*) which is linked to the corresponding plan via *prov:wasDerivedFrom*. The activity that generated an execution trace is recorded as *ep-plan:MultiActivity*. An execution trace is modelled using three main concepts based on the PROV model, namely *ep-plan:Entity*, *ep-plan:Activity* and *ep-plan:Agent* (see Figure 7).

Figure 8 illustrates how the individual parts of an execution trace are linked to plan elements describing steps and variables that are linked using the sub-plan mechanism. *ep-plan:EntityCollection* is used to represent execution instances of *ep-plan:MultiVariable* so individual entities from the execution trace of the sub-plan can be linked using the *prov:hadMember* relationship.

Figure 9 illustrates an example execution trace corresponding to a plan containing a communication constraint.

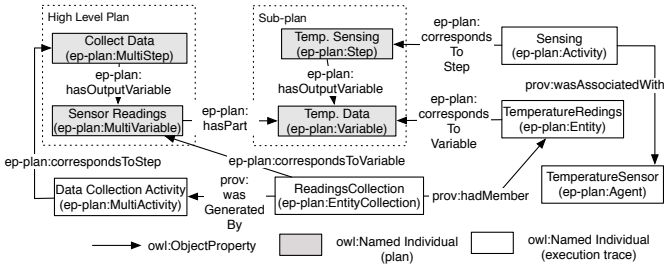

Fig. 8. An example partial description of a plan and a sub-plan and its corresponding execution trace. Links between instances representing plans and sub-plans (i.e. of type *ep-plan:Plan*) and their elements (e.g, steps) have been omitted to simplify the figure.

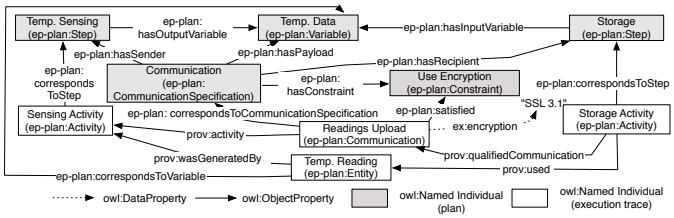

Fig. 9. An example partial description of a plan containing a communication constraint and its corresponding execution trace. Links between the instance representing the plan (i.e. of type *ep-plan:Plan*) and its elements (e.g, steps) have been omitted to simplify the figure.

ep-plan:Communication describes an execution instance of the *ep-plan:CommunicationSepecification*. Relationships *ep-plan:satisfied* and *ep-plan:violated* are used to indicate whether a particular instance of an execution trace satisfied or violated a constraint specification. If a constraint was associated with *ep-plan:Step* and *ep-plan:MultiStep* (not shown in Figure 9) such properties would link corresponding execution instances described as *ep-plan:Activity* and *ep-plan:MultiActivity*.

V. Example Queries

The EP-Plan ontology was implemented using the OWL 2 syntax[5] and documented with WIDOCO[6]. EP-Plan serialisations, documentation, examples, test datasets, and other relevant materials are stored and maintained in a public GitHub repository[7] under the Creative Commons 2.0 license[8]. The repository includes competency questions[9], sample datasets, and example SPARQL[10] queries[11] that cover EP-Plan's ability to retrieve information relating to elements used to describe plans (i.e. steps, variables, constraints, policies, rationales, objectives, responsible agents), their correspondence to the execution trace, and the ability to query linked plans and executions traces described at different abstraction levels.

[5]https://www.w3.org/OWL/
[6]https://zenodo.org/record/2576182/
[7]https://w3id.org/ep-plan
[8]https://creativecommons.org/licenses/by-nc-sa/2.0/
[9]https://github.com/TrustLens/EP-PLAN/blob/master/docs/cq/cq.csv
[10]https://www.w3.org/TR/sparql11-query/
[11]https://trustlens.github.io/EP-PLAN/examples/sample_datasets/iot/

For example, the SPARQL query illustrated in Figure 10 will return a list of all agents that are associated with some level of responsibility in *ex:HomeMonitoringHighLevelPlan* and all of its sub-plans.

```
SELECT DISTINCT ?agent
WHERE {
{ ex:HomeMonitoringHighLevelPlan a ep-plan:Plan;
       ep-plan:includesResponsibleAgent ?agent.}
UNION {
?subPlan ep-plan:isSubPlanOfPlan* ex:HomeMonitoringHighLevelPlan;
                        ep-plan:includesResponsibleAgent ?agent.}
}
```

Fig. 10. A query to retrieve all agents that can assume some responsibility for portions of a plan or its sub-plans.

The SPARQL query illustrated in Figure 11 will return a constraint specification associated with a step in *ex:HomeMonitoringHighLevelPlan* and the corresponding step execution activity which violated this constraint.

```
SELECT DISTINCT ?constraint ?activity
WHERE {
ex:HomeMonitoringHighLevelPlan ep-plan:includesConstraint ?constraint;
                               ep-plan:includesStep ?step.
?step ep-plan:hasConstraint ?constraint; a ep-plan:Step.
?constraint a ep-plan:Constraint.
?activity ep-plan:correspondsToStep ?step; a ep-plan:Activity.
?activity ep-plan:violated ?constraint.
```

Fig. 11. A query to retrieve execution trace activities that violated a constraint associated with a step.

The SPARQL query illustrated in Figure 12 will return the policy associated with *ex:HomeMonitoringHighLevelPlan* and any plan elements (e.g., constraints) that were included in the plan specification as a result of this policy.

```
SELECT DISTINCT ?policy ?planElement
WHERE {
ex:HomeMonitoringHighLevelPlan a ep-plan:Plan;
                ep-plan:includesPolicy ?policy;
      ep-plan:includesPlanElement ?planElement.
?policy a ep-plan:Policy; a ep-plan:Rationale.
?planElement  ep-plan:hasRationale ?policy.}
```

Fig. 12. A query to retrieve plan elements that were included as a result of some policy associated with the plan specification.

Such queries can then be used, for example, by an automated assessment service to assess user privacy risks associated with an IoT deployment - as proposed in our previous work [8].

VI. Related Work

A number of ontologies capture different aspects of plan specifications, their metadata and their association with provenance traces. D-PROV [13] and its successor ProvONE[12] extend PROV-O with a vocabulary for describing workflow specifications. Similarly, the Common Workflow Language[13] and its extension CWLProv [14] represent a community effort to generalise workflows and link them to their results. These models are designed for supporting descriptions of scientific

[12]http://purl.org/provone
[13]https://w3id.org/cwl/

workflows systems and do not capture concepts such as constraints, policies, agents that execute individual processes, etc. Mapping between these models and EP-Plan is possible at the level of connecting the descriptions of planned steps/processes and specifications of how data between such steps should be exchanged.

In previous work, including P-Plan extensions for domain specific applications such as social computation [15] and food safety [16], Markovic et al. explored the concept of constraints recorded as part of the plan description. EP-Plan brings this strand of work into a more generic, domain agnostic model. In EP-Plan we also expand the ability to associate plans with *policies* [11], [12].

While EP-Plan only contains a high level reference to policies and constraints, models such as the W3C recommendation ODRL [17] can be used with EP-Plan to define in detail the policies associated with a plan. Alternatively, the concept of policy defined in Dublin Core Terms[14] can be used to align ontological descriptions of high level policies such as GDPR[15]. Similarly, other vocabularies such as the W3C recommendation SHACL [18] and SPARQL Inferencing Notation (SPIN)[16] can be used with the plan description to specify constraints.

In the IoT domain, the W3C SSN ontology may be utilised to describe various characteristics of devices (e.g., their sensing capabilities). Plans described using EP-Plan are aligned with the SSN description of a device (*ssn:System*) by extending the *ssn:Procedure* followed by such a device.

VII. Conclusions & Future Work

In this paper we presented EP-Plan, a vocabulary for describing generic plan specifications and their corresponding execution traces. By proposing extensions to capture references to concepts such as constraints, agents, policies, objectives and rationales, we create opportunities for describing coherent cross-domain provenance records documenting *what was expected to happen* and *what actually happened* during system operation. EP-Plan's domain agnostic nature and ability to link across different levels of plan abstractions provides a powerful tool for interrogating and assessing a wide range of systems.

Our future work will focus on further validation and evaluation of EP-Plan as part of a prototype implementation for an automated privacy assessment framework for IoT deployments proposed in the *Trustlens* project [8]. This will generate additional examples detailing the use of EP-Plan in a domain specific application.

Acknowledgment

The work described here was funded by the award made by the RCUK Digital Economy programme to the University of Aberdeen (EP/N028074/1), a SICSA PECE travel award, the Defense Advanced Research Projects Agency with award W911NF-18-1-0027, the SIMPLEX program with award W911NF-15-1-0555 and from the National Institutes of Health under awards 1U01CA196387 and 1R01GM117097.

[14]http://dublincore.org/documents/dcmi-terms/#terms-Policy

[15]https://w3id.org/GDPRtEXT/gdpr#

[16]http://spinrdf.org/

References

[1] L. Moreau, P. Groth, J. Cheney, T. Lebo, and S. Miles, "The rationale of prov," *Journal of Web Semantics*, vol. 35, pp. 235 – 257, 2015.

[2] A. Gyrard, S. K. Datta, and C. Bonnet, "A survey and analysis of ontology-based software tools for semantic interoperability in iot and wot landscapes," in *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, Feb 2018, pp. 86–91.

[3] T. Lebo, S. Sahoo, and D. McGuinness, "PROV-O: The PROV ontology," Tech. Rep., April 2013. [Online]. Available: https://www.w3.org/TR/2013/REC-prov-o-20130430/

[4] D. Garijo and Y. Gil, "Augmenting prov with plans in p-plan: Scientific processes as linked data," in *Proceedings of the 2nd International Workshop on Linked Science*, vol. 951. CEUR Workshop Proceedings, 2012.

[5] V. A. F. Almeida, D. Doneda, and M. Monteiro, "Governance challenges for the internet of things," *IEEE Internet Computing*, vol. 19, no. 4, pp. 56–59, July 2015.

[6] Alliance for Internet of Things Innovation, "AIOTI Strategy 2017-2021," 2017. [Online]. Available: https://aioti.eu/aioti-strategy-2017-2021/

[7] N. Jacobs, P. Edwards, M. Markovic, C. Cottrill, and K. Salt, "Public sector internet of things deployments: Value, transparency, risks and challenges," in *Data For Policy*. Zenodo, 2019, doi to appear.

[8] M. Markovic, W. Asif, D. Corsar, N. Jacobs, P. Edwards, M. Rajarajan, and C. Cottrill, "Towards automated privacy risk assessments in iot systems," in *Proceedings of the 5th Workshop on Middleware and Applications for the Internet of Things*, ser. M4IoT'18. New York, NY, USA: ACM, 2018, pp. 15–18.

[9] M. Lefrançois, K. Janowicz, A. Haller, S. Cox, D. L. Phuoc, and K. Taylor, "Semantic sensor network ontology," W3C, W3C Recommendation, Oct. 2017, https://www.w3.org/TR/2017/REC-vocab-ssn-20171019/.

[10] D. Corsar, M. Markovic, and P. Edwards, "Capturing the provenance of internet of things deployments," in *Provenance and Annotation of Data and Processes*, K. Belhajjame, A. Gehani, and P. Alper, Eds. Cham: Springer International Publishing, 2018, pp. 196–199.

[11] J. Singh, T. F. J.-M. Pasquier, J. Bacon, J. E. Powles, R. Diaconu, and D. M. Eyers, "Big ideas paper: Policy-driven middleware for a legally-compliant internet of things," in *Middleware*, 2016.

[12] J. A. Padget and W. W. Vasconcelos, "Fine-grained access control via policy-carrying data," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 11, no. 1, pp. 31:1–31:24, Feb. 2018.

[13] P. Missier, S. Dey, K. Belhajjame, V. Cuevas-Vicenttín, and B. Ludäscher, "D-prov: Extending the PROV provenance model with workflow structure," in *5th USENIX Workshop on the Theory and Practice of Provenance (TaPP 13)*, 2013.

[14] F. Z. Khan, S. Soiland-Reyes, R. O. Sinnott, A. Lonie, C. Goble, and M. R. Crusoe, "Sharing interoperable workflow provenance: A review of best practices and their practical application in CWLProv," Dec. 2018, submitted to GigaScience (GIGA-D-18-00483).

[15] M. Markovic, P. Edwards, and D. Corsar, "Sc-prov: A provenance vocabulary for social computation," in *Provenance and Annotation of Data and Processes*, B. Ludäscher and B. Plale, Eds. Cham: Springer International Publishing, 2015, pp. 285–287.

[16] M. Markovic, P. Edwards, M. Kollingbaum, and A. Rowe, "Modelling provenance of sensor data for food safety compliance checking," in *Provenance and Annotation of Data and Processes*, M. Mattoso and B. Glavic, Eds. Cham: Springer International Publishing, 2016, pp. 134–145.

[17] S. Villata and R. Iannella, "ODRL information model 2.2," W3C, W3C Recommendation, Feb. 2018, https://www.w3.org/TR/2018/REC-odrl-model-20180215/.

[18] D. Kontokostas and H. Knublauch, "Shapes constraint language (SHACL)," W3C, W3C Recommendation, Jul. 2017, https://www.w3.org/TR/2017/REC-shacl-20170720/.