SENSE: Semantics-based Explanation of Cyber-Physical Systems

Deliverable 4.1 (v2): Semantic-Based Explainability Framework

Authors	:	Katrin Schreiberhuber, Mevludin Memedi, Fajar J. Ekaputra, Marta Sabou
Dissemination Level	:	Public
Due date of deliverable	:	31.07.2024
Actual submission date	:	31.08.2024
Work Package	:	4. Semantics-based Event Explainability
Type	:	Report
Version	:	2.0

Abstract

This deliverable is the second version of D4.1 from the SENSE project. This deliverable version contains a major update (v2) to the SENSE ontology with a detailed description of core classes and properties as well as example instances for these classes. The SENSE ontology is used as a basis for the other tasks within the work package. This update complements the existing content of type causality knowledge acquisition method (T4.2), the explanation generation framework (T4.3), and an initial literature study result of the personalized and actionable explanation approach (T4.4). We plan to elaborate more on the personalized and actionable explanation approach in the last version of the deliverable, scheduled for M30.

Project Funded by FFG – IKT der Zukunft Programme

Project Number: FO999894802 **Start date: 01.02.2023** **Duration: 30 months**

History

Version	Date	Reason	Revisited by
0.1	15.11.2023	Initial draft (v1)	FJE
0.2	28.02.2024	Ch.3,4,5	MM, KSch, MS
1.0	31.03.2024	Final review (v1)	FJE, MS
1.1	26.08.2024	Update on Ch.2 (v2)	KSch
1.2	29.08.2024	Abstract and Intro update (v2)	FJE
2.0	31.08.2024	Final review (v2)	MS

Author List

Project Partner	Name (Initial)	Contact Information
WU	Marta Sabou (MS)	marta.sabou@wu.ac.at
WU	Mevludin Memedi (MM)	mevludin.memedi@wu.ac.at
WU	Katrin Schreiberhuber (KSch)	katrin.schreiberhuber@wu.ac.at
WU	Fajar J. Ekaputra (FJE)	fajar.ekaputra@wu.ac.at

Table of Content

List of Figures

List of Tables

1 Introduction

The SENSE project aims to develop Explainable Cyber-Physical Systems (ExpCPS) that allow various interested stakeholders to understand system events. The project hypothesises that providing more transparency into CPS event explanations could boost efficiency, make them more user-friendly, and affect environmental sustainability.

The WP4 of the SENSE project aims to provide semantics-based event explainability over integrated CPS data, building on the integrated CPS data collected and integrated in WP3 of the project. To this end, WP4 contains four tasks as the following:

- *T4.1 The SENSE semantic model definition*. This task focuses on designing and developing a semantic model for knowledge-driven event explainability. The SENSE semantic model will consist of: (a) an overarching model for causality knowledge, (b) a model to describe CPS and their contexts, e.g., topology and environments, and (c) domain-specific models to capture specific data and information from use cases.
- *T4.2 Type causality knowledge acquisition*, focuses on employing automatic causal discovery methods to support domain experts in identifying type causality between CPS events.
- *T4.3 Explanation generation and ranking*, aiming to provide methods to generate and rank explanations for CPS events. The following steps to achieve our goals: (a) identification of actual causality knowledge between CPS events based-on type causality knowledge, (b) generation of causality paths based on actual causality for selected events, and (c) Ranking of the identified causality paths. First results of this task are published in a paper accepted at the 2024 Energy Informatics DACH+ Conference [2].
- *T4.4 Personalized and actionable explanation*. This task includes identifications and descriptions of user profiles to allow personalized explanations and represent them as a knowledge graph.

The work on WP4 will be conducted mainly by WU with the support and contributions from other SENSE project partners. The collaboration especially important on T4.1 towards the development of the SENSE semantic model. The list of involved partners is provided in Table 1.

Table 1: List of Involved project partners and members

Project Partners	**Project Members**
Wirtschaftsuniversität Wien	Katrin Schreiberhuber, Fajar J. Ekaputra, Marta Sabou, Mevludin Memedi
TU Wien	Gernot Steindl, Thomas Frühwirth, Muhammad Bilal
Siemens AG Österreich	Konrad Diwold, Daniel Hauer, Simon Steyskal
AEE INTEC	Christoph Moser
MOOSMOAR Energies OG	Wolfgang Prüggler

This deliverable is the second version of D4.1, which will be updated periodically (we plan to deliver the next version in M30) and includes progress and results from all four tasks in WP4. The rest of the deliverable is structured as follows: Section 2 reports the result of T4.1 on the SENSE semantic model, Section 3 describes the related work and experiments conducted on type causality knowledge acquisition of T4.2, Section 4 narrates our current result on the explanation generation and ranking algorithms, as well as its implementation and evaluation result in one of SENSE use case, and Section 5 outlines the result of our initial literature study on the topic of personalized and actionable explanation.

2 Semantic Model for Explainability

This task focuses on designing and developing semantic model necessary for knowledge-driven event explainability. The goal of the SENSE semantic model is to provide a solid basis for the development of (a) causality knowledge representation, (b) models to describe CPS and their contexts, e.g., topology and environments; and (c) domain-specific models to capture specific data and information from use cases.

2.1 Methodology

In this task, we followed the Linked Open Terms (LOT) methods [3] to develop the SENSE ontology.

LOT consists of four main steps: (i) *Ontology Requirement Specification* step, which resulted in a conceptual Ontology Requirement Specification Document (ORSD), (ii) *Ontology Implementation* step, where the requirements are implemented in a specific ontology language and evaluated using ontology evaluation tools, (iii) *Ontology Publication* step, where the ontology developed in previous step is published and documented in a web page, typically using publishing tools such as Widoco [4], and (iv) *Ontology Maintenance* step, where issues and bugs are collected, reported and used inputs to further develop and maintain the ontology in the future.

The focus of the T4.1 in this reporting period is on Step (i) and (ii). As part of the first step of Ontology Requirement Specification, we have conducted a series of workshops to gather inputs and requirements for the SENSE semantic model. The result of these workshops and the following analysis are collected in the following documents:

- The SENSE ORSD document[1], containing a high-level specification document of the SENSE ontology. This document contains links to the other documents: UC requirements document and the Modeling Data document.
- the UC requirements document[2], which reported the relevant competency questions, the origin and importance, and the subsequent answers to the questions.
- the Modeling Data (MODA) document[3]. This document contain the identified classes, properties, relations, as well as property of relations that are necessary for the SENSE ontology. Note that this document is not yet specific to specific modeling language.

In the Step (ii), we implemented the first draft of the ontology using Draw.io and Chowlk converter[5] within iterations. Furthermore, we evaluated the developed SENSE ontology using tools (e.g., OOPS [6]) and discussion with use case partners.

2.2 The SENSE Ontology v2

We propose an integrated data model as the foundation of an ExpCPS framework that is applicable across various CPS domains. The data model plays a key role in our approach as a means to facilitate the integration of diverse data sources for comprehensive system understanding and to enable root cause analysis. While different technologies can be used to implement the data model, we use semantic web technologies to exemplify the implementation of the

[1]"LOT_common_ORSD_v1.0.0.docx", the file is available on project repository
[2]"LOT_UC_Requirements_v1.0.0.docx", the file is available on project repository
[3]"LOT_MODA_v1.0.0.docx", the file is available on project repository

model and to visualise its structure. A semantic model serves as a structured representation of knowledge or information, designed to improve understanding by both humans and machines. It relies on an ontology, which acts as a schema for representing a specific domain [7]. Instantiating an ontology with data points results in the creation of a KG, where concepts and entities are uniquely identified using URI to ensure reusability. The semantic model builds upon the RDF [8] for creating a graph-based data structure. In the prototype, data points are published to the data model in RDF serialized in the Turtle format.

The data model consists of four parts, which focus on different aspects of the ExpCPS framework (i) System Topology (ii) Events and States (iii) Causal Knowledge (iv) User Context. In Fig. 2, the data model is shown as an ontology. Each concept or entity is identified by a URI. Prefixes are defined to abbreviate long URI strings The ontology implementation of the integrated data model is documented and published online[4], using WIDOCO [4] for the creation of a documentation of the Ontology.

We describe a motivating example of an ExpCPS from the area of SG to clarify our context. In this example, we introduce a public EV charging garage, which is directly connected to a local transformer. As the garage is a major energy consumer in the area, an operating envelope is imposed on the facility. This means, the facility operator needs to make sure that the electricity demand of the facility from the local grid stays within an agreed limit (within the operating envelope). Fig. 1 depicts the setup of the example use case. The garage has multiple chargepoints, where an EV can charge. It also has a battery installed, which can be used for peak shaving in times of high consumption. Multiple sensors are installed at various devices (indicated as gray circles), which take measurements of various observable properties, such as AP, SoC, OE.

During the operation of the garage, no envelope violation should occur, as the garage is supposed to regulate the use of its components autonomously, avoiding any violation of the imposed operating envelope. However, there are situations, where violations still happen. If they do, the facility operators as well as the distribution system operator (DSO) will request an explanation of why the violation has happened. An example explanation, which can be derived from the use case in Fig. 1 would be "`EnvelopeViolationState4` occurred due to `HighChargingState3` at `EVCharger1`, which could not be balanced out by the Battery as it was empty (`LowBatterySocState2`). High Charging Activity for multiple hours preceding the Violation has led to an empty battery (`HighChargingState1`)".

2.2.1 System Topology

The *System Topology* contains information about the system setup, all the devices in the system as well as the sensors, which provide a continuous data stream of system measurements at runtime. The topology information specifies the positions of the devices and their physical and logical connections. To represent the topology of a system, we use four core concepts: platform, sensor and observable property. The terms are derived from the SOSA Ontology[5], a "lightweight general-purpose ontology to represent the interaction between entities" in CPSs [9]. Additionally, we introduce the concept of sensor type, as a separate class to define the properties of sensor types independent of a specific instance of a sensor.

A **Platform** is an entity that hosts other entities, particularly sensors, and other platforms (e.g., EV charger, battery, garage). Platforms are devices or facilities within a system, where sensors are located.

[4] `http://w3id.org/explainability/sense#`
[5] `https://www.w3.org/TR/vocab-ssn/`

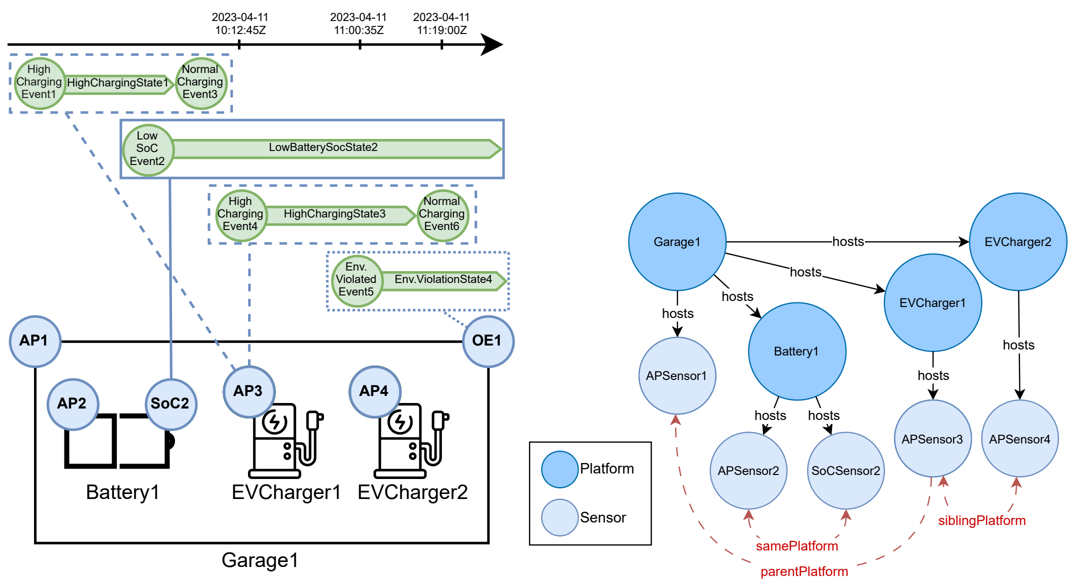


Figure 1: Left: EV garage as motivating example, showing ActivePower(AP), StateOfCharge (SoC), Operating Envelope (OE) sensors in blue, events and states connected to the sensors in green.
Right: topological representation of the motivating example according to the ontology.

A **Sensor** is a device or agent, which collects data measurements of an observable property (e.g., AP). Sensors are hosted by a platform, which means a sensor is located at/within a Platform and its measurements correspond to this platform (e.g., `EVCharger1` hosts `APSensor3` in Fig. 1). Each sensor stores information on how to access its sensor measurements (e.g., access token to a timeseries database).

A **SensorType** is a concept of a sensor, defining the purpose of a specific type of sensor and its relation to other concepts. Each sensor has a sensor type, which defines the type of measurements it can take and what types of states are possible at the sensor.

An **Observable Property** is an observable quality (property, characteristic) in the system. Each sensor observes an observable property. (e.g., `APSensor3` observes `ActivePower`).

By using these concepts (Platform, Sensor, Sensor Type, Observable Property), the system setup can be represented in sufficient detail to facilitate event explainability. It enables the description of which sensors are located at which platform, what a sensor measures, where to find measurement data and how platforms are connected to each other.

2.2.2 Events and States

The *Events and States* data contains any information connected to the event detection and state derivation process. It contains event detection methods for each event type that is implemented to detect events. Additionally, it stores a priori information about the event-to-state type mapping between event types and state types. At runtime, detected events as well as states that are derived from these events are collected and stored in this module. As an example, `HighChargingEvent4` is detected at `"2023-04-11 10:12:45Z"` by `APSensor3` hosted by `Charger1` in the motivating example from Fig. 1.

Event: Each event (e.g., `HighChargingEvent4`) is stored in the data base, including added semantics about the event:

- *eventType* of the event that was detected (e.g., `HighChargingEvent`)
- *procedure* that is responsible for the detection of the event (e.g., `HighChargingDetection-`

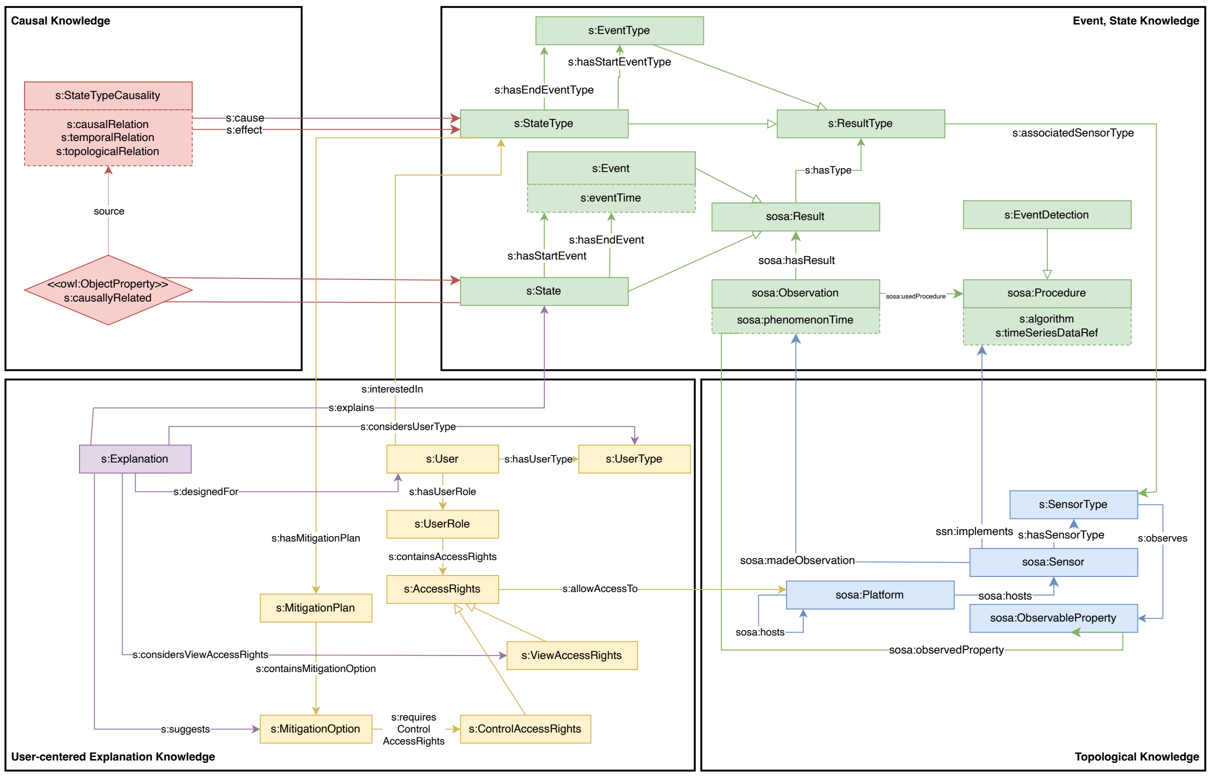


Figure 2: Integrated Data Model represented as an ontology

`Query`). A procedure can be a query, a model, or any function that is responsible for the event detection.

- *sensor* where the observation was made (e.g., `APSensor3`)
- *timestamp* of the event (e.g., `"2023-04-11 10:12:45Z"`)
- *observableProperty* that was observed to detect the event (e.g., `ActivePower`)

State: The *State Derivation Module* is responsible for deriving states from detected events using the event-to-state mapping. A new state (e.g., `HighChargingState3` in Fig. 1) is stored in the database including the following semantics:

- *stateType* of the state (e.g., `HighChargingState`)
- *startEvent* that has started the state (e.g., `HighChargingEvent4`)
- *endEvent* that has ended the state (e.g., `NormalChargingEvent6`); if the state has not been ended yet, it is also possible that there is no endEvent related to the state (e.g., see `EnvelopeViolationState4` has no endeEvent (yet))
- *observableProperty* that was responsible to detect the event (e.g., `ActivePower`)

Users can query for potential root causes of specific system states. More information on the identification of a root cause for a trigger state is provided in Sections 4.

Event-to-State mapping: Each event type represents a state transition between state types, which is defined a priori (e.g., `HighChargingEvent` starts `HighChargingState`, and `Normal-ChargingEvent` ends `HighChargingState`). The state derivation module uses this information combined with topology data to identify which state has been changed at which sensor by a certain event (e.g., `HighChargingEvent4` at `APSensor3` starts `HighChargingState3` at

`APSensor3` as a result of the event-to-state mapping indicating `HighChargingEvent` starts `HighChargingState`).

2.2.3 Causal Knowledge

The *causal knowledge* data is a description of causal relations between state types. Each causality relation consists of a cause state type and an effect state type as well as three parameters (causal, temporal, topological) that give detailed information on the nature of the causal relation. We rely on the definition in the Causal and Temporal Relation Scheme (CaTeRS) [10] for causal and temporal relations. An example of such a relation is:

> `HighChargingState` (**cause state type**)
>
> `causes` (**causal**) an
>
> `overlapping` (**temporal**)
>
> `EnvelopeViolationState` (**effect state type**) at
>
> `parentPlatform` (**topological**)

Each part of the causal relation is described in more details below.

cause state type and effect state type: A state type is the concept of a state. It has a description of what characteristics this type of state exhibits (e.g., `HighChargingState` is a state when an `EVCharger` charges more than 50kW). A state type is not related to a specific time or place. It is a concept of a state that can occur at runtime.

causal: The causal relation defines a more detailed view of the type of causality as compared to a simple ”caused by” relation. There are three types of causal relations that can occur between two states ($stateA \rightarrow stateB$):

- *cause:* If stateA occurs, stateB most probably occurs as a result.
- *enable:* If stateA does not occur, stateB most probably does not occur (not enabled to occur).
- *prevent:* If stateA occurs, stateB most probably does not occur as a result.

By providing a more detailed description of a causal relation, intricate relations between state types can be represented, such as ”`LowBatterySoCState` prevents `BatteryDischargeState`, while `BatteryUnusedState` (i.e. the non-existence of `BatteryDischargeState`) enables `EnvelopeViolationState`”.

temporal: The temporal relation captures the temporal aspects of a relation between two states. There are two options for temporal relations ($stateA \rightarrow stateB$) that showed to be relevant for representing causal knowledge:

- *before:* stateA starts and ends before stateB.
- *overlaps:* stateA starts before stateB, but ends after stateB has started.

topological: The topological relation considers the logical/hierarchical relation between two states as a constraint for the causal relation to hold. We consider a hierarchical representation of platforms and their sensors. A state is always associated with one or more sensors, hosted by a platform. In Fig 1, the topology representation of the motivating example is depicted on the right side. Platforms are shown in light blue, while sensors are shown in dark blue. Three types of topological relations are defined as causal constraints. each of the relations is described below and exemplified in red in Fig. 1:

- *samePlatform:* stateA and stateB are associated with a sensor on the same platform (e.g.,`SoCSensor2` and `APSensor2` are associated to the same platform `Battery1`).
- *parentPlatform:* stateA is associated with a sensor on a platform that is hosted by the platform of stateB (e.g., `APSensor1` is associated with `Garage1`, which is the parent platform of `EVCharger1`. `APSensor3` is associated with `EVCharger1`, forming a `parentPlatform` relation from `APSensor3` to `APSensor1`). Note that this relation is the only non-symmetric topological relation.
- *siblingPlatform:* stateA is associated with a sensor on a platform that is hosted by the same platform as the platform associated with stateB (e.g.,`APSensor3` is associated with `EVCharger1` and `APSensor4` is associated with `EVCharger2`. Both platforms are hosted by `Garage1`, so the two sensors are hosted by `siblingPlatforms`).

The definition of these relations can be extended if needed for specific domains or use cases.

A thoroughly crafted causal knowledge dataset is crucial for the explanation engine to provide meaningful and correct explanations of states. This knowledge not only provides information about causality, but it also provides information on which types of states are relevant. Therefore, it serves as an indication to create corresponding event detection procedures in the event detection module of the ExpCPS framework.

2.2.4 User Context

The *user context* module contains any information connected to a user of the SENSE System. Each user has a certain role in the system, preferences, access rights, and system states they are interested in, which influence the explanation the will get. In order to create a user-centered and actionable explanation, all of these features need to be known to the system.

User: Each user, who is interested in receiving an explanation from the SENSE System is stored in the knowledge based, together with their features:

- *UserType* defines a user in terms of their preferences for a specific explanation medium, or technicality level of an explanation (e.g. lay user vs technical user)
- *UserRole* defines a user in terms of the user context inside the system a user role defines the access rights of a user within the system. View Access Rights define what Platforms a user can know about, thus which platforms are included in an explanation for this user, while Control Access Rights define the Mitigation Options, which are presented to the user (e.g. unplugging an EV can only be a valid mitigation option for the owner of the EV).

MitigationPlan: Is a plan, which is defined for a certain state type, and it lists a set of options to avoid/mitigate a certain type of state. Each option requires specific control access rights to be conducted.

Explanation: An explanation stores the user-centered version of a causal path, which depends on the User Type, *ViewAccessRights* and *ControlAccessRights* of a user. Each explanation is designed for a user of the system and explains a *state* instance. Based on the user and the causal path, mitigation options are presented to a user.

3 Type Causality Knowledge Acquisition

3.1 Towards causality identification problem: From time series to causality relations

Understanding how a set of variables are related is becoming and important aspect when developing an information system that makes decisions using large amount of observational data. Having knowledge about causal relationships between variables help in comprehending the dynamics behind complex systems. Data-driven methodologies using statistics and machine learning (ML) allow processing vast amounts of data and applying algorithms to find causal relationships among variables with the aim of establishing causality.

Causality of $X \rightarrow Y$ ($\rightarrow$ meaning X causes Y) can be defined if and only if an intervention or manipulation in X affects Y, where X is known as independent, or cause variable and Y is known as dependent or effect variable. In complex systems, the effects are known events and are time specific. One approach is to calculate correlation coefficient between the variables. Nevertheless, the measure of correlation provides the degree of association and does not consider the directionality of the relationship. Therefore, more advanced measures should be used to provide the directionality among the variables.

Identifying causality relations among multiple variables in complex systems is crucial for understanding the underlying characteristics and dynamics of the system. Such exploratory data analysis as suggested by [11] would help in getting insights on how the different system components interact and influence each other over time, which in turn could be helpful making informed decisions. Another benefit is gaining knowledge on cause-and-effect relationships that can be used for predicting future states or events within the system and regulate the system more effectively. However, the task of inferring causality relation among variables comes with a set of challenges. In complex systems, interactions between variables can be nonlinear and there can be time delays between the variables. There can also be confounding variables that may influence both the cause and effect and can complicate the process of causal inference. Additionally, causal relationships among variables within a system can change over time and failing to capture such changes may result in misleading conclusions.

Time series data of complex systems describe traces of systems over time and can be used as inputs to quantitative methods for inferring causality relations between them. Time series can be derived as observations recorded during experiments or sensors collected over time. The goal of time series analysis for causal identification in complex and dynamic systems is to statistically and reliably estimated causal links, including their time lags [12]. The causality identification problem is related to the challenge of extracting causal relationships between variables based on observed time-series data. For instance, consider a set of variables denoted as X_1, X_2, ..., X_n, where each X represents a time-varying variable within a complex system M (figure 3, [1]). The main aim of the analysis is to determine whether there exists a causal relationship between a variable X_j (the cause) and a target variable X_k (the effect). The causality identification problem involves detecting whether changes in the values of X_j influence changes in the values of X_k. The two aims of such analysis would be to detect general causal interactions among the components of M, including time lags and to quantify the strength of causal interactions within M in a well-interpretable way.

The challenge during causality identification from time series data is in estimating a function that represents the causal relationship and determining the appropriate time lag or delay between the cause and effect. Various methods such as Granger causality tests and structural equation modeling, which are statistical methods, and Transfer Entropy (TE), which is an

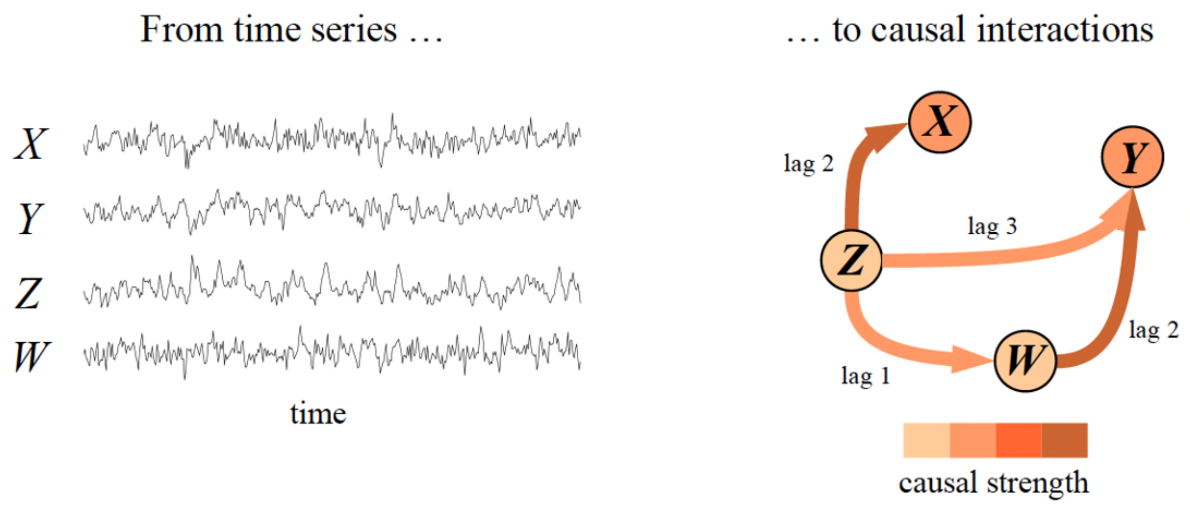


Figure 3: The process of analyzing different time series within a complex system to extract causality relationships between them [1]

information-theoretic method, are employed to address this problem.

3.2 Related Work

Time series are present in various forms and can be generated in different domains, for instance in healthcare through monitoring systems using devices and sensors, industry through predictive maintenance systems, and other application domains by using data such as images, acoustic signals, etc.

In literature, many studies have focused on identifying causality problem and many techniques have been introduced. The very first work on causality inference was presented by Clive Granger in 1969 [13] where the Granger causality test was introduced. This method has been widely used studies dealing with, but not limited to, econometrics time series data. The idea with the method is to determine if a time series can be used to predict the future values of another time series. This method was to study brain connectivity using neuroimaging time series [14]. In the paper by Sugihara et al. [15] causality methods were used to infer causalities within climate systems based on analysis of time series data. The paper addressed the challenges of processing time series where there are nonlinear and non-stationary dynamics. When determining causation in time series there is a need to model the delayed causation between them and this need inspired the development of vector autoregression, which involves analyzing previous data samples with current ones to determine delayed causal links [16]. One drawback of this method is that when applied in datasets with high dimensionality it can lead to low detection power. In addition, when dealing with time series data one important aspect is to consider methods that are robust to detect non-linear interactions and operate without any priori model. Granger causality test does not address nonlinearity in time series.

Another method that has been applied in literature is TE. TE is like Granger causality [17]. Nevertheless, Granger causality is not good for modelling non-linearity in the time series. This drawback of Granger causality is related since it is based on linear models and is not appropriate for nonlinear systems. TE has been proved as effective measure of capturing dynamical features among different components in time series. TE has been applied in different application domains. TE has been used by in a real-world dataset consisting of financial time series to characterize the information flow among different stocks [18]. Additionally, it has been applied in neuroscience for identifying causal interactions in brain networks [19]. TE has been used as a metric to detect presence of nonlinearity in health monitoring systems [20]. Even TE has its own disadvantages. It is more challenging to be automated, computationally intensive, and more complex to interpret [21]. To overcome these issues and at the same time account for non-linearity in time series some methods have been proposed utilizing ML methods. For instance, Tank et al. [22] suggested a new, extended Granger-based model using Structured Multilayer Perceptrons or Recurrent Neural Networks. Similarly, Nauta et al. [23] has proposed a new method for discovering causal relationships in time series data using attention-based Convolutional Neural Networks. Rossi et al. [24] presented a software tool implementing an architecture of timed-delayed neural networks (TDNNs) for causality-detection in physical time dependent systems. TDNNs has been shown to perform well and does not require an extensive amount of data to provide robust conclusions.

3.3 Causality detection algorithms implemented in the SENSE project

In this section of the report, we will provide a summary of the methods that were implemented within the SENSE project to infer causal relations from time series. The dataset that we used was about a smart grid use case and simulated using Siemens Bifrost simulation engine. The dataset contained data about a month with a range of time series representing different properties of a smart grid system. Some of the variables that were included in the dataset were: `TRAFO_SECONDARY_VOLTAGE@TRAFO_BUILDING`, `VOLTAGE_ANGLE3P@GRID_NODE`, following the naming format `VARIABLE@SYSTEM_COMPONENT`. The methods were applied on the time series selected based on expert knowledge and tested to quantify the causality relations among them.

The aim was to infer causal relations between different properties in a cyber-physical system e.g. smart grid. Such systems are very complex in nature with a large number of interactions between its properties. A challenge would be to design a custom ML model (e.g. neural network) that captures key aspects of the underlying system processes and that can generalize to different situations that could occur within a smart grid. In addition, the process for designing and validating an ML model requires sufficient domain knowledge and understanding of the mechanisms within a smart grid system to correctly specify the underlying model. Not being able to map the underlying relations within the system would also impact the possibility to check the validity of our methods [15]. An ML approach would also be time consuming to design since it would require mapping of the time series of the different properties of a system to different events. This type of work would need sufficient labeled data for training and validation and involvement of domain experts.

An alternative that we have undertaken in the project was to employ model-free approaches, which typically rely on statistical learning [25]. By doing this, our models would not need to focus on specific dynamics of a target system (in our case a smart grid system) but instead examine the characteristics ("Information flow") among the time series of the system and cross map them. In view of this, we rationalized for the use of the following model-free approaches, including Granger causality and TE, that have been commonly used in literature to infer causal relations in time series data in different fields of studies. As a complement to the statistical methods, the pre-trained TDNN tool [24] was implemented and tested.

Below follows an overview of the methods that were implemented.

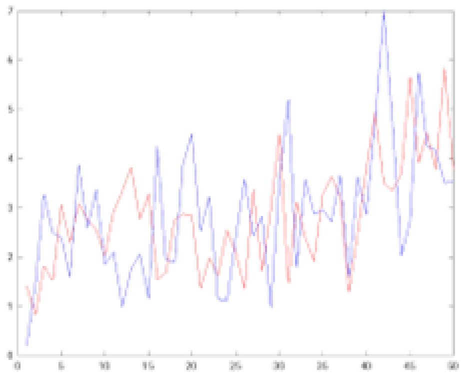


Figure 4: An example with two time series X (red-colored) and Y (blue-colored)

3.3.1 Method 1: Granger causality

Granger causality is based on prediction. If a signal X is causal for another Y, then past values of X should contain information that helps to predict Y better than merely using the information contained in past values of Y. Alternative definition: "If discarding X reduces the power to predict Y then X contains unique information about Y and thus can be concluded that X Granger-causes Y". The presence of this relation between X and Y can be referred as "X Granger causes Y".

Mathematically, this relationship can be formalized by a vector autoregressive (VAR) model where future value of Y is modeled as a linear combination of historical values of X and Y (figure 4). Each VAR coefficient is then tested for its significance whether it is helpful for predicting Y via t-tests followed by testing the final model via an F-test. If any coefficient is significant then we can conclude "X Granger causes Y". The concept of including an additional variable for better prediction of another variable is related to the linear regression model where an independent variable is declared significant if the full model predicts the dependent variable better than the model without this variable. The underlying assumption is that the time series should not be stationary and that their properties should not depend on the time they are observed. For instance, time series with trends or seasonality are not stationary since the former will affect the values in the time series at different times.

A Granger Causality test would return an output where

- 0 – indicates that the null hypothesis of noncausality should not be rejected. In other words, there is not enough evidence to suggest that the time series X Granger-causes the time series Y.
- 1 – indicates rejection of the null hypothesis of noncausality, which means that there is enough evidence to suggest that the time series X Granger-causes the time series Y.

3.3.2 Method 2: Transfer Entropy (TE)

TE is a measure used to quantify the directed information flow between two time series [26, 27]. TE provides means for quantifying how much information from the past of one time series can be used to predict the future of another time series. In other words, TE describes the uncertainty of an information in time series to quantify the directed information flow from one time series to another.

In contrast to Granger causality method which is regression-based, TE is an information-theoretic method and can capture both linear and non-linear dependencies between time series, making it suitable for analysis of causality in complex systems. The larger the value of TE, the more information is transferred from one time series to another, suggesting a stronger influence of a time series on another one. A value of 0 indicates no influence or information transfer between the time series.

3.3.3 Method 3: Pre-trained TDNN

Disadvantages when using Granger causality and TE:

- Can give contradictory results and can perform better in certain specific application than others
- Not appropriate for problems with many time series. Granger causality and TE cannot represent the nonlinearity when using multiple time series at the same time.

We have used the toolbox implemented by Rossi et al. [24] and has the following functions:

1. Causality Detection: performs a statistical test to evaluate if the variance of a time series is statistically different from the variance of another time series. If a p-value is smaller than a threshold (e.g. 0.05), a time series influences another time series.
2. Time Horizon Detection: measures the time horizons of the influence, that is the time difference between the influencing event of X and the influenced event in Y.

Next, we will provide technical details about implementation of the three methods. Methods 1 and 3 run on Matlab and method 2 runs on Python environments. Each method requires the data in specific formats. For instance, methods 1 and 3 require the data to be formatted as a Matlab catalogue with ".mat" extension. The method 2 requires the data to saved in a CSV format. To run methods 1 and 3 the Matlab toolboxes likl Econometrics, Deep Learning Toolbox, Statistics and Machine Learning Toolbox, should be installed on the computer running the code. The method 2 requires the computer to have Jupyter environment to be set up along with libraries such as Panda and PyIF to be installed.

3.4 Evaluation of the methods

In this section we will provide details on the experiments that were performed (Table 3). For all the experiments simulated data provided by our industrial partner Siemens were used.

The following results were derived for experiment 1. When hypothesis 1 was tested we found out that the methods 1 and 2 could detect causality. Nevertheless, for other hypotheses the results were mixed showing contradicting causality relations and/or two-way causality relations (that is the time series cause each other in both ways).

For experiment #2, which focused on specific events that were pre-identified through observations in a visualization tool and during the time starting from the event and an hour later, the results are displayed in Table 4.

The causality results from the methods are different to each other. In other words, when one method detects causality, the other methods do not detect any causality. The results are not either consistent within the methods.

After segmenting the time series 2 hours before the event and an hour after the event according to experiment #3 the results showed no changes. Lastly, as per experiment #4 the

Experiment#	Goal	Description of tests and hypotheses
1	To test causality detection algorithms using time series measured during the whole time.	Set of cases were derived from a project member who has expert knowledge in smart grids. Data segments: full period of the measurements that is one month. Hypotheses: • PV Power Production causes Batter State of Charge • Car Charging Demand causes Grid Demand • Car Charging Demand from Fast Charger causes Battery State of Charge (Reduction) • Grid Demand causes Battery Demand (or the other way around)
2	To test causality detection algorithms focusing on specific events happening in the smart grid system	Eight events in the dataset were identified using a visualization tool developed as part of another task in the project. The tool visualized the different time series and the time when a violation in the system occurred. A set of cases were derived by a colleague. Data segments: To improve the power of the methods to detect causality we decided to focus on a time segment related to the events. We decided to consider data for analysis from the start of the event till an hour after the event. Hypotheses: • Grid Demand causes Total Charging • Total Charging causes Battery Charge/Discharge
3	Test the methods with different time intervals.	The assumption was that when an event happens the characteristics and links between different time series are changed and reflected before the actual event. For this reason, we decided to focus on the same scenarios as in experiment #2 but include 2 hours of data before the event. Same scenarios and hypotheses were tested as in experiment #2.
4	Test refined versions of the methods.	The methods were refined by preprocessing time series before using them as inputs to methods. The assumption was the time series had non-linearity features. To eliminate nonlinear trends from the time series a low (6)-order polynomial filter was fitted to them. Same scenarios and hypotheses were tested as in experiment #2.

Table 3: Experiments and hypotheses tested.

	Grid demand causes Total charging	Total charging causes Grid demand	Both cause each other	No causality	Not relevant
GC	2	2	3	1	
TE	2	3	0	2	
TDNN	3	2		1	1

Table 4: Results for the 8 system events for each method and combination of causality.

time series were initially pre-processed using a filter to remove the non-linearity features in them and then used as inputs the three methods. Similarly, no changes in the results were observed.

3.5 Conclusions and Future work

The findings from the conducted experiments yield both insightful and complex results.

In experiment 1, the detection of causality through hypothesis 1 demonstrated promising results, especially, for methods 1 and 2. However, the presence of mixed and contradictory causality relations for other hypotheses, including two-way causality, highlights the complex nature of the relationships within the time series. Experiment 2 focused on specific pre-identified events. Again, significant divergence in causality results among the employed methods was observed. There was a lack of consistency within methods and cases where one method detected causality while others did not establish reliable causality links. Contrary to expectations, experiments 3 and 4, which involved time series segmentation around events and pre-processing time series by removing non-linearity features, did not yield observable changes in the results. These results suggest that temporal proximity to events might not significantly impact the causality relations, at least within the investigated time series frame.

As future work, one might consider the following steps. A more nuanced examination of causality during specific events could provide valuable insights. For instance, discussing the cases with domain experts may help in ensuring the existence and magnitude of causality relations between time series during a specific time frame. Additionally, testing data from diverse real-world data sources (e.g. smart buildings in cooperation with the industrial project partner AEE INTEC) could offer a comprehensive understanding of the relationships between time series, which in turn could shed light into validity and reliability of the methods. Finally, there would be a need to conduct extensive validation and robustness testing of the causality detection methods against known ground truth scenarios to provide confidence in their applicability and reliability.

4 Explanation Generation and Ranking

Explainability of a system is the ability to explain its behavior to the user. In a world, where increasingly complex systems are helping users in their everyday lives, explainability is becoming a crucial feature to be implemented in any system. A system, which can explain its decisions and reason about the root cause of a situation increases user trust, reduces maintenance time and costs and even helps users to make the right decisions when interacting with the system.

By definition, an explanation seeks to make a situation, object or event more understandable. However, creating understandable explanations is a tricky task. There is no one-fits-all solution on building an explanation that makes sure the explanation seeker improves their understanding of a certain situation. To achieve understandable explanations, multiple factors have to be considered, such as the knowledge of the explanation seeker, the context they are in as well as the actual reason for a situation to occur. In a Cyber-Physical System (CPS), an explanation of a situation or decision of the system can be created by analysing the system data. However, such a trace-based explanation is only one possible type of explanation that could satisfy user needs.

We are proposing an integrated explanation engine for Cyber-Physical, which can derive user-centered explanations from system topology data, time series sensor inputs and causality knowledge from domain experts. For a good representation of causality, we propose a multi-faceted causality input. This more intricate representation allows us to create more accurate and precise explanations, catering explanations to user's needs and context.

In this chapter, we are presenting the current version of the explanation engine in the SENSE project. In figure 5, the envisioned workflow from input data to explanations is shown. The first part, getting from raw time series data to Events, is part of Workpackage 3, in Task 3.3 and wil not be elaborated further here. The focus of this task (explanation generation and ranking is the center of the figure, where Events are converted to Causal Paths with the help of additional input, such as topology information of the system as well as Event type causality knowledge. The output of this task would be a list of causal paths, which constructs the basis for user-centered and actionable explanations (to be tackled in T 4.4). In this first version, we have focused on the SeeHub PoC. It contains a public garage, which offers 8 EV charging stations. The garage further contains a battery as well as a PV System, which help to balance the power needs of the electric vehicles charging at the facility. The local grid operator has imposed an operating envelope on the facility operator of the garage, determining the minimum and maximum capacity the garage is allowed to demand or feed into the local grid. This is necessary to prevent the local power transformer from overloading. If the envelope is violated, an explanations is required for the facility operator of the garage. This is the explanations we are aiming for.

In section 4.1, we will discuss the literature on events and causality that forms the basis of our approach. In section 4.2, we will focus on the methodology of the approach to get from Events to Causal Paths and will then show first results of applying this approach to the SeeHub PoC. Finally, we will discuss our findings and future improvements of the work in section 4.3.

4.1 Related Work

Explanations and Explainability has been discussed and researched in many domains. In the context of the SENSE project, we will focus on explainability in Cyber-Physical Systems as well as explainability in AI. While explainable AI is not the focus of the project, there are many parallels between the two domains, especially since some form of AI, either in the form of

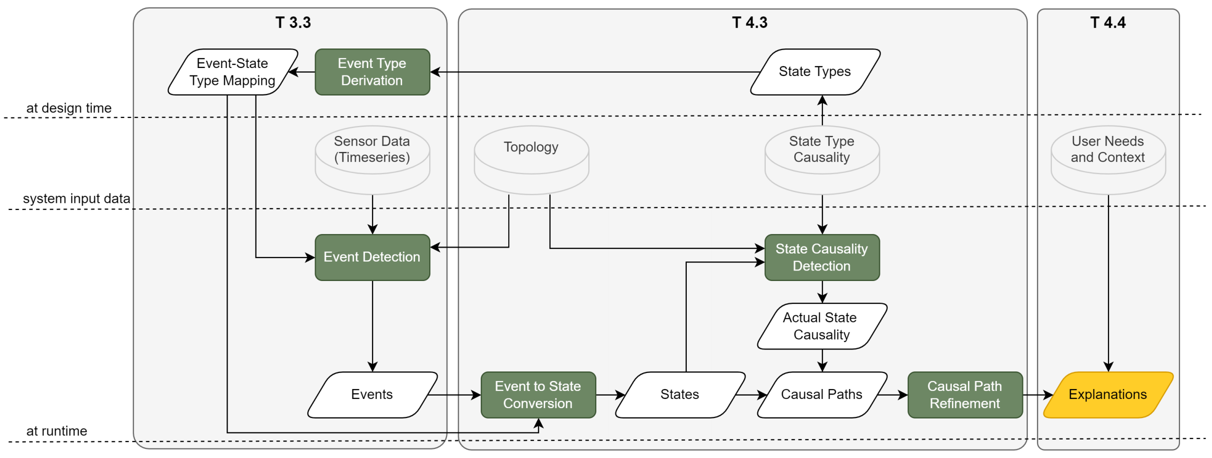


Figure 5: Workflow from input data to Explanations

decision support or data analysis processes, is implemented in almost every CPS. Therefore, we will analyse the research of explainability in both domains before we conclude with an analysis of the most important aspects to consider from either of them.

As AI becomes a regular part of our daily lives, the fact that most AI models are ”black-boxes” in terms of how they come to their decisions has resulted in increased research on explainable AI. Making AI models explainable can help in increasing user trust as well as establish accountability in model decisions. However, current Machine Learning models mostly rely on statistical inference, which is very different from human-level reasoning. While statistics is based on probability values based on pre-seen data, human reasoning is based on causal models, where we use our knowledge of cause and effect relations between events to explain a situation or decision. For a user to comprehend the decisions of a model, it is important to enhance statistical inference with causal knowledge. These causal cognitive models have only recently been ”algorithmitized” to enable mathematical exploration of causal relations [28].

On another note, research on explainability has shifted from mechanistic explanations to user-centric explanations, which are context-aware and consider the comprehensibility of an explanation [29]. While mechanistic explanations provide provenance information and justifications that are objectively valid, they might be insufficient to meet the user’s needs, especially in the face of increasingly complex systems that need explaining. User-centered explanations are thus an extension of the provenance-enabled explanations as they incorporate domain-knowledge and user’s context to generate comprehensible explanations of the functioning of a system.

Depending on the user, the type of explanation that is suitable for them might differ. In [30], an overview explanation types including their definitions and sufficiency conditions for each explanation type is presented. The authors argue that explanations for users need to include multiple types of explanations as each type serves a different purpose.

4.2 Methodology and initial Results

We have established a workflow, where causal relations between events in the system can be derived based on their position in the system, temporal occurrence as well as their event types. Descriptions of data types, classes and properties of the Semantic Data Model are described in detail in section 2.

As Input Data for the Semantic Explanation Engine, the following Data Sources are required:

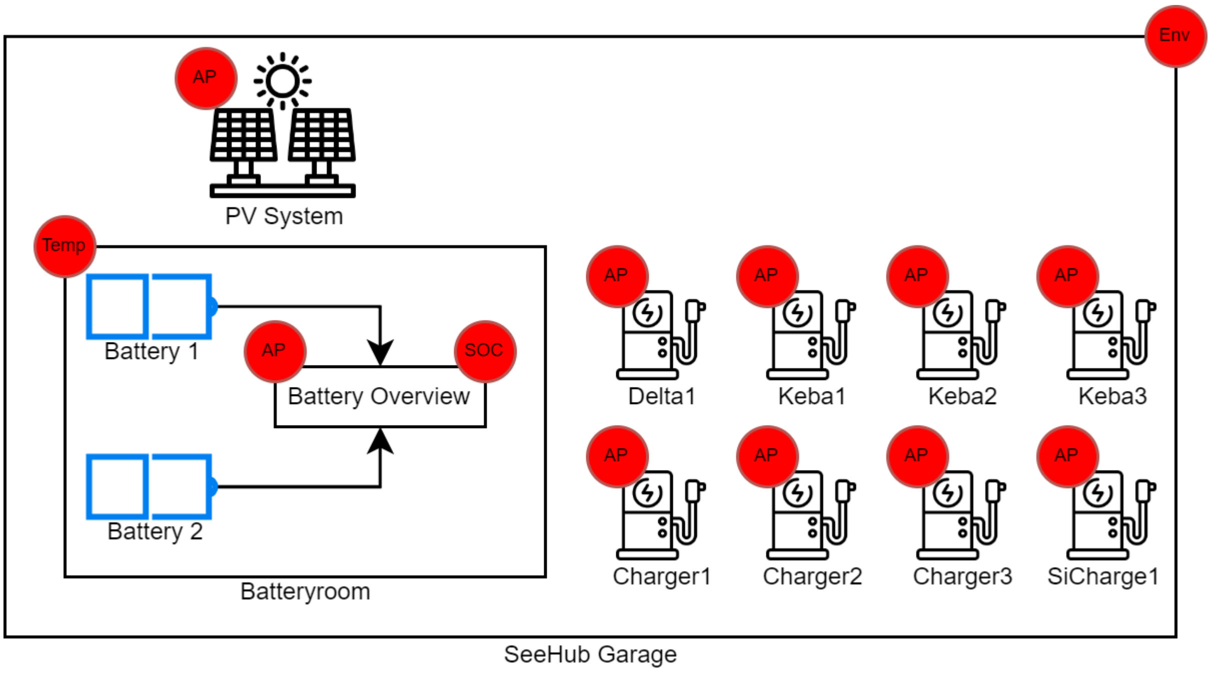


Figure 6: SeeHub topology

- **Topology Data** topology data contains static data about the topology of the system. It describes the platforms, sensors and connections between them, what type of properties the Sensors measure.
- **State Types and State Type Causality** These Datasets contain a list of potential States, which are of interested within the system. A State is an occurence of interest, which happens over a period of time. Each StateType has a type of Sensor it can be associated with. It is possible that a StateType can be associated with multiple types of sensors.
- **Event Detection Data Stream** At runtime, a continuous data streams of detected events provides the basis for the creation of states and further to detect causal paths for explanations.

We have used data from the SeeHub use case for creating and testing a first version of the Explanation Generation Framework. In figure 6, the topology of the SeeHub use case is shown. In this use case, there is an operating envelope imposed on the facility operator, which defines a range of maximum electricity power demand or feed-in of the facility to the grid. Violations of the envelope range need to be detected and explained. A more detailed description of the use case can be found in Deliverable 2.1, where the SENSE use cases are defined in detail. Components of the system (i.e. PV System, Chargers, Batteries) can have Sensors attached to them, which measure some observable property (shown as red circles). Any component within the outline of another component is considered a sub-component of the bigger system. This setup constructs the topology input for the engine.

StateType	Definition	associated SystemType
Envelope Violation	The active power consumed/produced within the platform is exceeding the envelope limits imposed by the operatin envelope of the platform	EnvelopeViolation_Garage_Sensor
High Charging	active power consumed by the EVcharger is exceeding the operating envelope of the platform above.	AP_EVCharger_Sensor
Low Battery SOC	state of charge of the battery is within the lowest 5% ever recorded	SOC_Battery_Sensor
Battery Dip	active power provided by the battery has dropped for a time instance	AP_Battery_Sensor
Battery Discharge Loading	active power provided by the battery is increasing, but has not reached its high point yet	AP_Battery_Sensor
High Charging Difference	active power consumed by the EVcharger is increasing very fast	AP_EVCharger_Sensor
Battery Unused	battery is neither consuming any active power nor providing active power to the system	AP_Battery_Sensor

Table 6: SeeHub State Types

State Types and State Type Causality was defined based on the Use Case requirements and example explanations. In table 6, the list of state types that are of interest in the SeeHub system are listed. Each state type has a definition as well as a type of sensor it can be associated with.

In addition to the state types in the system, we need to define the relations between them. The causal relation between types of states are defined in the state type causality table. It contains the information about how events are connected to each other in as many facets as possible. In the CaTeRS scheme [10], a temporal and causal relation scheme was introduced, considering not only a simple ”caused by” relation, but dividing the relation further into the temporal aspect as well as the causal aspect of the relation. We are using the terms for temporal and causal relations as defined in [10]. For temporal relations, there are four options (explained in the form $eventA \rightarrow eventB$):

- **before** eventA starts and ends before eventB
- **overlaps** eventA starts before eventB, but ends after eventB has started
- **contains** eventA starts before eventB and ends after eventB has ended
- **identity** eventA starts and ends at the same time as eventB

Furthermore, there are three options for causal relations [10]:

- **cause** If eventA occurs, eventB most probably occurs as a result.
- **enable** If eventA does not occur, eventB most probably does not occur (not enabled to occur).
- **prevent** If eventA occurs, eventB most probably does not occur as a result.

Cause	causal	temporal	topological	Effect
High Charging Difference	enables	overlaps	siblingPlatform	Battery Discharge Loading
Battery Discharge Loading	causes	overlaps	parentPlatform	Envelope Violation
High Charging	causes	before	siblingPlatform	Low Battery SOC
Low Battery SOC	causes	before	parentPlatform	Envelope Violation
Battery Dip	causes	overlaps	parentPlatform	Envelope Violation
High Charging	causes	overlaps	parentPlatform	Envelope Violation
High Charging	enables	overlaps	samePlatform	High Charging Difference

Table 7: State Type Causality input

Additionally, we have added a third category to the type causality definition: topological. This relation indicates the topological relation between two events in the system. These types are defined:

- **samePlatform** eventA and eventB are associated with a sensor on the same platform
- **parentPlatform** eventA is associated with a sensor on a platform that is hosted by the platform, where eventB is detected
- **siblingPlatform** eventA is associated with a sensor on a platform that is hosted by the same platform as the platform associated with eventB is.

In table 7, the state type causality for the SeeHub use case is shown. This table is derived based on domain expert knowledge for now. Future work could focus on different options to derive this information in a more automated way.

Based on the input data on topology, state types and state type causality, the explanation generation algorithm can derive a set of causal paths for any state that requires an explanation. In a first step, actual causality is derived from type causality through reasoning. The algorithm to create actual causality relations is desrcibed in algorithm 1. In the next step, a causal path needs to be found in the system through exploratory search for actual causality relations. The function to find a full causal path is described in algorithm 2. It is a recursive function, querying the causal path for actual causal relations between events up to the point, when no more causes are detected in the system. The result is a tree of causal paths, where all branches lead to a potential root cause. The final root cause in the system is not necessarily the actual root cause, but it is the last step of the cause, where reliable data is available. Any further causes would be events happening outside of our defined system and thus we do not have data about them.

4.3 Conclusion and Future Improvements

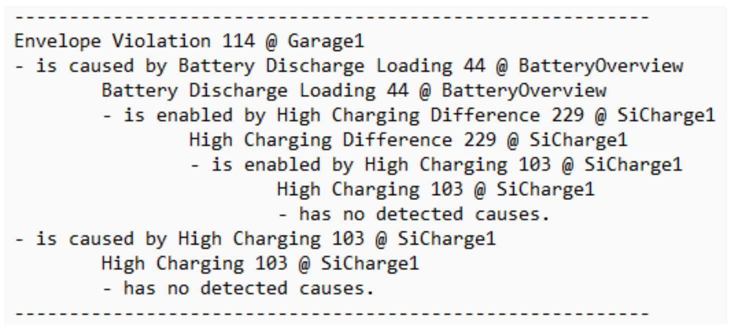
We have presented the SENSE explanation engine in its current version. We have established its usefulness in the SeeHub use case, which is a small real-life e-charging facility in Aspern. Future work will focus on implementing the engine for other use cases in the smart grid and smart building domain. Additionally, the refinement of the explanations and the evaluation of their correctness will be tackled in the next months within the project. We have written up and refined this approach in a paper accepted at the 2024 Energy Informatics DACH+ Conference [2].

Algorithm 1 derivation of state causality from state type causality

```
 1: for each $typeCausality$ in $StateTypeCausality$ do
 2:     for each $state$ in the data stream that has $eventType = typeCausality.Effect$ do
 3:         $effectSensor \leftarrow$ sensor associated with $state$
 4:         $effectPlatform \leftarrow$ platform hosting $effectSensor$
 5:         $effectTimeInterval \leftarrow$ time between $start$ and $end$ of $state$
 6:         $causePlatforms \leftarrow$ list of platforms which have $typeCausality.topological$
                relation to $effectPlatform$
 7:         $causeSensors \leftarrow$ list of sensors hosted by $causePlatforms$
 8:         $causeTimeInterval \leftarrow$ limits for $start$ and $end$ of a potential $causeState$
                according to $typeCausality.temporal$
 9:         for each $state$ in the data stream fulfilling $causeTimeInterval$ associated with
    $causeSensors$ do
10:             add Causality relation ($causeState$,$typeCausality.causal$,$effectState$)
11:         end for
12:         Check for cause event types in the potential systems
13:         Return for now
14:     end for
15: end for
```

Algorithm 2 causal path identification

```
1: function findCause(effectState)
2:     query $causeStates$ in the relation ($causeState, causalrelation, effectState$)
3:     for each $state$ in $causeStates$ do
4:         save $state$ as a cause of $effectState$
5:         findCause(state)
6:     end for
7: end function
```



```
-----------------------------------------------------------
Envelope Violation 114 @ Garage1
- is caused by Battery Discharge Loading 44 @ BatteryOverview
        Battery Discharge Loading 44 @ BatteryOverview
        - is enabled by High Charging Difference 229 @ SiCharge1
                High Charging Difference 229 @ SiCharge1
                - is enabled by High Charging 103 @ SiCharge1
                        High Charging 103 @ SiCharge1
                        - has no detected causes.
- is caused by High Charging 103 @ SiCharge1
        High Charging 103 @ SiCharge1
        - has no detected causes.
-----------------------------------------------------------
```

Figure 7: example of a causal path detected by the explanation engine

5 Personalised and Actionable Explanation

Explanations in the SENSE-architecture aim to be catered to a user's needs. Additionally, we will also focus on the access rights and context of a user who is requesting an explanation of the system. All the information surrounding the user will be represented in the semantic model. In a recently submitted paper to the Neurosymbolic AI journal about the importance of ontologies in explainable AI, Confalonieri and Guizzardi [31] explain the importance of considering a user's context to create good explanations.

User-centered explanations are becoming a increasingly important in the era of explainable AI. The importance of a user-centered design of explanations has already been mentioned in 2007 by [32]. Explanations are a broadly used term, but explanations can have different aims, such as transparency, scrutability, increase of trust, effectiveness, persuasiveness, efficiency or satisfaction. In a user studies on movie recommendations and explanations of the recommendations, they have found that explanation features need to be tailored to the user and the context.

In another paper, which has been accepted to PerCom 2024 [33], a user-centered explanation engine has been proposed in the smart home domain. They have developed a system, where the user profile as well as the user state and role of the user is considered at the time of occurence of an event. All of these layers are important aspects which will be considered in the personalised and actionable explanations of the SENSE system.

On another note, the definition of a *good* explanation remains an open topic until today. Since explanations are generated with different goals in mind (as mentioned above), their evaluation is tends to be application-specific as well. In [34], an explainability factsheet is proposed to evaluate an explanation in a structured way. The evaluation is to be done in five dimensions:

- functional - type of problem to be addressed
- operational - type of interaction with the end user
- usability - how comprehensible an explanation is to the end-user
- safety - robustness and security of an approach, echting if there is information leaks and consistent information
- validation - process to evaluate and prove the effectiveness of the explanation

While an explanation should aim to perform well in all dimensions, there is usually a trade-off happening between some of the dimensions. Which dimension is more important is finally an application-specific decision to be made.

In a first step to tackle this research problem, we have already integrated user context into the SENSE Ontology, as described in section 2.2.4.

References

[1] J. Runge, “Detecting and quantifying causality from time series of complex systems,” 2014.

[2] K. SCHREIBERHUBER, F. J. EKAPUTRA, M. SABOU, D. HAUER, K. DIWOLD, T. FRÜHWIRTH, G. STEINDL, and T. SCHWARZINGER, “Towards a state explanation framework in cyber-physical systems,” in *DACH+ Energy Informatics 2024, Lugano, Switzerland, Proceedings*. ACM SIG Energy, 2024.

[3] M. Poveda-Villalón, A. Fernández-Izquierdo, M. Fernández-López, and R. García-Castro, “LOT: An industrial oriented ontology engineering framework,” *Engineering Applications of Artificial Intelligence*, vol. 111, p. 104755, May 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197622000525

[4] D. Garijo, “Widoco: a wizard for documenting ontologies,” in *The Semantic Web–ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II 16*. Springer, 2017, pp. 94–102.

[5] S. Chávez-Feria, R. García-Castro, and M. Poveda-Villalón, “Chowlk: from uml-based ontology conceptualizations to owl,” in *European Semantic Web Conference*. Springer, 2022, pp. 338–352.

[6] M. Poveda-Villalón, A. Gómez-Pérez, and M. C. Suárez-Figueroa, “Oops!(ontology pitfall scanner!): An on-line tool for ontology evaluation,” *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 10, no. 2, pp. 7–34, 2014.

[7] L. Ehrlinger and W. Wöß, “Towards a definition of knowledge graphs.” *SEMANTiCS (Posters, Demos, SuCCESS)*, vol. 48, no. 1-4, p. 2, 2016.

[8] D. Tomaszuk and D. Hyland-Wood, “Rdf 1.1: Knowledge representation and data integration language for the web,” *Symmetry*, vol. 12, no. 1, 2020. [Online]. Available: https://www.mdpi.com/2073-8994/12/1/84

[9] K. Janowicz, A. Haller, S. J. Cox, D. Le Phuoc, and M. Lefrançois, “Sosa: A lightweight ontology for sensors, observations, samples, and actuators,” *Journal of Web Semantics*, vol. 56, pp. 1–10, 2019.

[10] N. Mostafazadeh, A. Grealish, N. Chambers, J. Allen, and L. Vanderwende, “CaTeRS: Causal and Temporal Relation Scheme for Semantic Annotation of Event Structures,” in *Proceedings of the Fourth Workshop on Events*. San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 51–61. [Online]. Available: https://aclanthology.org/W16-1007

[11] J. W. Tukey *et al.*, *Exploratory data analysis*. Reading, MA, 1977, vol. 2.

[12] J. Runge, P. Nowack, M. Kretschmer, S. Flaxman, and D. Sejdinovic, “Detecting and quantifying causal associations in large nonlinear time series datasets,” *Science advances*, vol. 5, no. 11, p. eaau4996, 2019.

[13] C. W. Granger, “Investigating causal relations by econometric models and cross-spectral methods,” *Econometrica: journal of the Econometric Society*, pp. 424–438, 1969.

[14] A. K. Seth, A. B. Barrett, and L. Barnett, “Granger causality analysis in neuroscience and neuroimaging,” *Journal of Neuroscience*, vol. 35, no. 8, pp. 3293–3297, 2015.

[15] G. Sugihara, R. May, H. Ye, C.-h. Hsieh, E. Deyle, M. Fogarty, and S. Munch, "Detecting causality in complex ecosystems," *science*, vol. 338, no. 6106, pp. 496–500, 2012.

[16] E. E. Leamer, "Vector autoregressions for causal inference?" in *Carnegie-rochester conference series on Public Policy*, vol. 22. North-Holland, 1985, pp. 255–304.

[17] L. Barnett, A. B. Barrett, and A. K. Seth, "Granger causality and transfer entropy are equivalent for gaussian variables," *Physical review letters*, vol. 103, no. 23, p. 238701, 2009.

[18] X. Mao and P. Shang, "Transfer entropy between multivariate time series," *Communications in Nonlinear Science and Numerical Simulation*, vol. 47, pp. 338–347, 2017.

[19] M. Rubinov and O. Sporns, "Complex network measures of brain connectivity: uses and interpretations," *Neuroimage*, vol. 52, no. 3, pp. 1059–1069, 2010.

[20] J. Nichols, M. Seaver, S. Trickey, M. Todd, C. Olson, and L. Overbey, "Detecting nonlinearity in structural systems using the transfer entropy," *Physical Review E*, vol. 72, no. 4, p. 046217, 2005.

[21] B. Lindner, L. Auret, M. Bauer, and J. W. Groenewald, "Comparative analysis of granger causality and transfer entropy to present a decision flow for the application of oscillation diagnosis," *Journal of Process Control*, vol. 79, pp. 72–84, 2019.

[22] A. Tank, I. Covert, N. Foti, A. Shojaie, and E. B. Fox, "Neural granger causality," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 8, pp. 4267–4279, 2021.

[23] M. Nauta, D. Bucur, and C. Seifert, "Causal discovery with attention-based convolutional neural networks," *Machine Learning and Knowledge Extraction*, vol. 1, no. 1, p. 19, 2019.

[24] R. Rossi, A. Murari, L. Martellucci, and P. Gaudio, "Netcausality: A time-delayed neural network tool for causality detection and analysis," *SoftwareX*, vol. 15, p. 100773, 2021.

[25] A. R. Coenen, S. K. Hu, E. Luo, D. Muratore, and J. S. Weitz, "A primer for microbiome time-series analysis," *Frontiers in genetics*, vol. 11, p. 310, 2020.

[26] J. T. Lizier, M. Prokopenko, and A. Y. Zomaya, "Local information transfer as a spatiotemporal filter for complex systems," *Physical Review E*, vol. 77, no. 2, p. 026110, 2008.

[27] M. Lindner, R. Vicente, V. Priesemann, and M. Wibral, "Trentool: A matlab open source toolbox to analyse information flow in time series data with transfer entropy," *BMC neuroscience*, vol. 12, pp. 1–22, 2011.

[28] J. Pearl, "The seven tools of causal inference, with reflections on machine learning," *Communications of the ACM*, vol. 62, no. 3, pp. 54–60, Feb. 2019. [Online]. Available: https://dl.acm.org/doi/10.1145/3241036

[29] S. Chari, D. M. Gruen, O. Seneviratne, and D. L. McGuinness, "Directions for Explainable Knowledge-Enabled Systems," Mar. 2020, arXiv:2003.07523 [cs]. [Online]. Available: http://arxiv.org/abs/2003.07523

[30] S. Chari, O. Seneviratne, D. M. Gruen, M. A. Foreman, A. K. Das, and D. L. McGuinness, "Explanation Ontology: A Model of Explanations for User-Centered AI," Oct. 2020, arXiv:2010.01479 [cs]. [Online]. Available: http://arxiv.org/abs/2010.01479

[31] Roberto Confalonieri and Giancarlo Guizzardi, “On the Multiple Roles of Ontologies in Explainable AI,” *Neurosymbolic Artificial Intelligence Journal*, 2023. [Online]. Available: https://www.neurosymbolic-ai-journal.com/paper/multiple-roles-ontologies-explainable-ai

[32] N. Tintarev and J. Masthoff, “Effective explanations of recommendations: user-centered design,” in *Proceedings of the 2007 ACM conference on Recommender systems*, 2007, pp. 153–156.

[33] M. Sadeghia, L. Herbolda, M. Unterbuscha, and A. Vogelsanga, “SmartEx: A Framework for Generating User-Centric Explanations in Smart Environments.”

[34] K. Sokol and P. Flach, “Explainability fact sheets: A framework for systematic assessment of explainable approaches,” in *Proceedings of the 2020 conference on fairness, accountability, and transparency*, 2020, pp. 56–67.