Enhancing TrUStAPIS Methodology in the Web of Things with LLM-generated IoT Trust Semantics

Davide Ferraris[1], Konstantinos Kotis[2], and Christos Kalloniatis[2]

[1] Network, Information and Computer Security Lab, University of Malaga, Malaga, Spain
ferraris@uma.es
[2] Department of Cultural Technology and Communication, University of the Aegean, Mytilene, Greece
{kotis,chkallon}@aegean.gr

Abstract. In the Internet of Things (IoT) there are ecosystems where their physical 'smart' entities virtually interact with each other. Often, this interaction occurs among unknown entities, making trust an essential requirement to overcome uncertainty in several aspects of this interaction. However, trust is a complex concept, and incorporating it in IoT is still a challenging topic. For this reason, it is highly significant to specify and model trust in early stages of the System Development Life Cycle (SDLC) of IoT-integrated systems, thus enhancing the aforementioned task. TrUStAPIS is a requirements engineering methodology recently introduced for incorporating trust requirements during IoT-based system design. The scope of this paper is to provide an extension of TrUStAPIS by introducing IoT trust semantics compatible with the W3C Web of Things (WoT) recommendations generated with the assistance of Large Language Models (LLMs). Taking advantage of LLMs as a tool for integrating and refining existing methodologies, in this paper we present our work towards a revision of the TrUStAPIS methodology. In this work, we contribute a new conceptual model and a refined JSON-LD ontology that takes into account IoT trust semantics, providing eventually a valuable tool for software engineers to design and model IoT-based systems and services.

Keywords: Trust, Internet of Things (IoT), Web of Things (WoT), Large Language Model (LLM), JSON-LD

1 Introduction

The Internet of Things (IoT) is a network of physical "smart" objects that exchange data with other devices over the Internet. While communication and interoperability are by definition the crux of the Internet of Things, the emergence of custom or proprietary solutions results in devices that cannot talk to each other due to the heterogeneity in data interchange mechanisms [1].

To integrate these disparate devices, developers must work with a growing set of protocols, serialization formats and API specifications. This results in repetitive, non-scalable and error-prone work that is difficult to automate [2].

While technologies like OpenAPI and AsyncAPI largely solve this problem in the context of Web APIs, they fall short for describing networks of non-HTTP and multi-protocol devices, and do not consider different modes of interaction based on their meaning in the physical world.

To solve these problems, the W3C Web of Things (WoT) recommendation group [3] works on providing standardized building blocks that make use of JSON Schema.

JSON Schema is used for validating descriptions of network-facing capabilities of physical entities called Thing Descriptions, and to model and describe data sent by IoT consumers and producers in a multi-protocol manner.

A solution for developing an IoT entity since the first phases of the System Development Life Cycle (SDLC) has been implemented by Ferraris et al. [3], namely TrUStAPIS. In this solution, which focuses in the requirements phase of the SDLC, JSON has been considered as crucial for eliciting requirements. However, such solution must be updated considering the evolution of the IoT ecosystem and we believe that we can use Large Language Models (LLMs) to assist humans performing this task.

The paper is structured as follows. Section 2 presents background knowledge which is necessary to understand the proposed approach and related work about WoT, JSON-LD and LLMs. Then, in Section 3, we motivate the research question. In Section 4, we describe the proposed approach, whereas in Section 5, we schematically represent the steps performed towards improving the TrUStAPIS methodology with LLMs' assistance. Next, in Section 6, we analyze the LLMs outputs comparing them to the expected ones. Finally, in Section 7, we conclude the paper and discuss future work.

2 Background Knowledge and Related Work

In this section, we provide a brief overview of the background knowledge required to comprehend the paper proposal. Specifically, we discuss the topics of trust and IoT, followed by an exploration of Requirements Engineering within the context of IoT, focusing on aspects such as security, privacy, and trust. Additionally, we present existing literature that implements technologies like WoT, JSON-LD and LLMs.

2.1 Trust and IoT

IoT is a network of interconnected things. Roman [1] states that the goal of IoT is to enable things to be connected anytime, anyplace, with anything and anyone, ideally using any network path and any service. It is expected that

[3] https://www.w3.org/WoT/

these entities will often have to interact with each other in uncertain conditions. Mechanisms to solve this lack of information are needed and trust can help address this need to overcome uncertainty. Trust is a concept that is difficult to define "because it is a multidimensional, multidisciplinary and multifaced concept" [4]. Jøsang [5] defines trust as personal and subjective, for McKnight [6] trusting someone means to depend on him, no matter the consequences. Typically, there are two entities (at least) involved in a trust interaction, one is the trustor (the entity which places trust) and the other is the trustee (the entity in which trust is placed). According to Hoffman et al. [7] and Pavlidis [8] trust is strongly dependent on other domains like privacy, identity and security. From these definitions, Ferraris et al. [9] stated that, in an IoT entity development, it is important to centrally consider trust and related domains such as usability or identity. Trust is related to each of them and they cover all the aspects that can increase and guarantee trust in an entity. Moreover, it is fundamental to consider all these domains for the whole SDLC of an IoT entity. For this reason, the authors developed a framework called K-Model, shown in Figure 1.

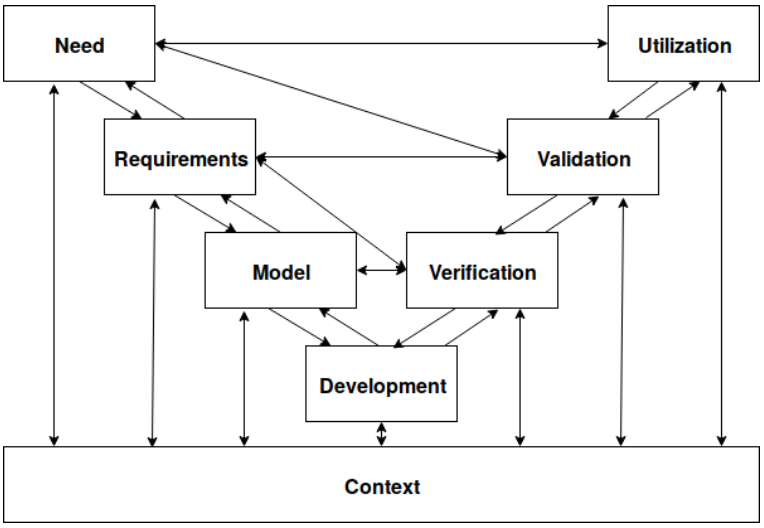


Fig. 1. K-Model [9].

In the K-Model are represented the whole phases of the SDLC, from the conception of the idea (i.e., need) to the final utilization of the developed IoT entity. The entity must be verified and validated before to be used by the customers. However, the development must follows a rigorous design (i.e., model) performed after the requirements elicitation. In this paper we will focus on this last described phase. Howevere, in the literature, several methodologies have been designed to fulfill this fundamental phase of the SDLC.

2.2 Requirements Engineering

As we mentioned earlier, the field of requirements engineering has seen extensive utilization following the emergence of Goal-Oriented methodologies. The seminal work developed by Yu [10] introduced concepts such as actors, objectives, and dependencies. An extension of Yu's work has been SI* [11]. This work further developed these concepts, particularly in the realm of security and trust, while TROPOS [12], based on the I* framework, aimed to facilitate all design activities within the SDLC. Inthe following years, Mouratidis and Giorgini expanded the TROPOS methodology with Secure Tropos [13], explicitly delineating actor ownership of services and their provisioning capabilities. Then, considering trust-related domains, Rios et al. [14] emphasized the significance of integrating privacy considerations during requirements engineering to ensure trust in negotiation processes. Finally, Mavropoulos et al. [15] introduced a methodology for eliciting security requirements for IoT adopting JSON, asserting that implementing JSON format could automate the elicitation process, thereby enhancing the analysis of extensive IoT networks.

Ferraris et al. [3] took all these works into considerations in order to advance beyond such paradigms by intertwining trust with other domains such as security, usability, and identity. Moreover, the authors introduced traceability to establish connections among the elicited requirements belonging to different domains. Such feature was not considered in previous methodologies. Therefore, they proposed a JSON-based template and a conceptual model to aid developers in eliciting comprehensive requirements encompassing elements pertinent to TrUStAPIS.

2.3 TrUStAPIS Methodology

In this paper, we aim to refine the TrUStAPIS methodology [3] by integrating Large Language Models (LLMs) to aid in requirements elicitation across diverse domains such as trust, security, or privacy. In fact, TrUStAPIS is an acronym compiled as follows:: **Tr**ust, **U**sability, **S**securi**t**y, **A**vailability, **P**rivacy, **I**dentity and **S**afety. In this section, we provide a brief overview of this methodology to establish a solid foundation for readers.

The elicited requirements align with the needs identified in the initial phase of the K-Model (see Figure 1 and adhere to the IEEE 830-1993 specification [16]. Each requirement is tailored to its domain characteristics. For instance, trust requirements consider aspects like transitivity and asymmetry, while privacy requirements encompass factors such as anonymity and confidentiality. It's worth noting that certain characteristics, such as confidentiality, may be pertinent to multiple domains (e.g., privacy and security). Moreover, a requirement may consist of one or more sub-requirements, a feature that helps developers in specifying requirements with greater detail. Another important aspect is that TrUStAPIS enables developers not only to elicit domain-specific requirements, but also to establish traceability among them. Furthermore, it accommodates

dynamic aspects related to IoT through the incorporation of context considerations.

Each requirement contains several key elements such as an *actor*, an *action* and a *goal*. All of these elements depend on a *context*. They are structured in the following way:

- **Actor**: It mainly represents entities, whether human or IoT-based. The actor is responsible for fulfilling or requesting the fulfillment of goals. Actors may assume various roles, such as trustor and trustee, considering the trust domain.
- **Action**: Tasks performed by actors, potentially associated with specific measures that aid in requirement modeling and subsequent verification and validation.
- **Goal**: The ultimate objective driving the identification of requirements, achieved by actors through appropriate actions. Goals are inherently tied to specific capabilities of IoT entities within the relevant domains.
- **Context**: A dynamic aspect closely tied to the domain of requirements, encompassing characteristics belonging to trust, usability, security, availability, privacy, identity, and safety. The context may vary based on the IoT paradigm and is influenced by environmental factors and the scope of the goal.

In addition, to facilitate requirement elicitation, we have proposed a JSON template and a conceptual model incorporating the aforementioned elements. In this paper, we expand this methodology upon the previous work by enhancing these tools and introducing IoT trust semantics generated with the assistance of LLMs. Such semantics are proposed for consistency within the WoT.

2.4 Web of Things and JSON-LD

The WoT is a framework aimed at fostering interoperability among various IoT platforms and application domains. It seeks to address the challenges of IoT fragmentation by providing a standardized approach to connect devices and applications across different ecosystems seamlessly. Essentially, WoT enables devices and services to communicate and interact with each other regardless of their underlying technologies or protocols [4].

JavaScript Object Notation Linked Data (JSON-LD) [17], or JSON for Linked Data [18], is implemented within the WoT framework to facilitate the serialization of Linked Data. Linked Data refers to a method of structuring data that enables interlinking and semantic interpretation, allowing machines to understand the relationships between different pieces of information. JSON-LD extends the widely-used JSON format to incorporate Linked Data principles, providing a straightforward means of representing data in a format that is both human-readable and machine-understandable.

[4] https://www.w3.org/TR/wot-architecture

Moreover, JSON-LD finds application within the WoT paradigm, which endeavors to mitigate IoT fragmentation by leveraging and extending established web technologies. Given the widespread adoption of IoT, the substantial volume of data exchange between devices and application domains underscores the necessity for interoperability. For example, a weather application necessitates communication with a traffic application, requiring data interpretation and interoperability across both domains. Nonetheless, to enhance interoperability, the Semantic Web of Things (SWoT) [19] paradigm advocates for the incorporation of semantics, based on Semantic Web technology, into implementations.

However, not all stakeholders possess familiarity or proficiency with Semantic Web standards such as RDF/XML or OWL. To overcome this issue, Elsayed et al. [20] proposed WOTJD [5] for WoT, employing JSON-LD. WOTJD assists IoT users in overcoming the primary challenges of data interoperability within WoT by facilitating the design and integration of WoT applications, IoT data parsing and annotation, and the linkage of domains leveraging domain knowledge expertise. A case study illustrating the resolution of interoperability issues between smart cars and the weather domain through a mobile application is presented, accompanied by performance evaluation. Experimental findings demonstrate the efficiency of the WOTJD framework relative to sensor data size.

We took these works into consideration in order to expand TrUStAPIS methodology into the WoT paradigm and proposing ontologies based on JSON. However, we decided to put LLMs in the equation benefiting of the high amount of data processing in order to find a suitable solution to reach our goal.

2.5 Large Language Model (LLM)

Large Language Models such as ChatGPT [6] or Gemini [7] have demonstrated remarkable outcomes in numerous Artifical Intellige (AI) applications. Research has shown that these models implicitly capture vast amounts of factual knowledge within their parameters, resulting in a remarkable performance in knowledge-intensive applications. Basically, in order to work with LLMs it is possible to train them, fine tuning them or simply apply prompt engineering. These tasks can be performed in sequence or it is possible to consider only a subset of them.

According to these possibilities, Alivanistos et al. [21] focused on prompting as probing, a multi-step methodology amalgamating various prompting techniques to construct knowledge bases from LLMs. In our study, we adopt a similar strategy.

Then, Li et al. [22] concentrated on task-specific enhancements for relation prediction using LLMs. They generate a sentence for each subject-relation-object triple, masking tokens relevant to the object entity, and train the LLM to predict these tokens. Additionally, they employ prompt elicitation.

[5] https://github.com/wotjd
[6] https://chat.openai.com/
[7] https://gemini.google.com/app

In their work, Pitis et al. [23] utilized few-shot prompts to construct prompt ensembles. They adapt classical boosting algorithms iteratively to enhance prompts.

Jiang et al. [24] proposed an interesting works about which LLM to choose and to know in order to utilize it in research. According to this work, we have performed our decision.

However, all these works presented the potentialities of prompt engineering in order to apply fine tuning to a chosen LLMs. We took these works into consideration in order to make a step forward and apply these ideas to requirements engineering. So far, to the best of our knowledge, in the state of the art there is not any work which implements LLMs to generate IoT trust semantics in JSON-LD. We will now explain the motivation related to our work.

3 Motivation

As previously mentioned, WoT extends the capabilities of IoT by leveraging web technologies to promote seamless integration and interoperability among IoT devices and services. It aims to establish a unified ecosystem where IoT devices can communicate, exchange data, and interact securely using standardized web protocols and interfaces, ultimately leading to more accessible and user-friendly IoT solutions.

Enhancing WoT framework in the TrUStAPIS methodology augments trust, security, and privacy aspects within the IoT ecosystem. TrUStAPIS emphasizes the significance of trust throughout the entire SDLC, ensuring that trust relationships between IoT devices, services, and users are established based on predefined requirements and trust models. By integrating TrUStAPIS, developers can boost security mechanisms such as authentication, authorization, encryption, and secure communication protocols, thereby safeguarding IoT devices and data from cyber threats and unauthorized access. Additionally, TrUStAPIS addresses privacy concerns by defining requirements for data minimization, user consent, and transparent data handling practices, thereby ensuring compliance with privacy regulations and respecting user preferences.

Furthermore, TrUStAPIS facilitates traceability and connectivity between requirements, promoting interoperability by defining standardized formats, interfaces, and protocols for exchanging trust, security, and privacy-related information among diverse IoT devices and platforms. Its support throughout the entire system development lifecycle, from requirements elicitation to decision-making and automation, enables developers to manage trust, security, and privacy considerations effectively at each stage of IoT solution development, thereby ensuring continuous monitoring and improvement of system security and privacy practices.

Incorporating LLMs into this process can significantly aid in handling vast amounts of data quickly, particularly in requirements engineering tasks. However, it's essential to assess the effectiveness of LLMs and determine whether human intervention is necessary for checking and modifying LLMs responses. Furthermore, the integration of prompt engineering and fine-tuning processes

for LLMs offers significant potential for enhancing the JSON template from TrUStAPIS into the JSON-LD format within the WoT framework. LLMs can be useful in order to create more structured and semantically meaningful Linked Data representations. By fine-tuning LLMs on specific tasks related to JSON-LD generation and interpretation, developers can improve the accuracy and efficiency of JSON-LD serialization and parsing processes. This enhancement not only streamlines data exchange and interoperability within the WoT ecosystem but also enhances the overall semantic richness and expressiveness of JSON-LD representations, enabling more effective integration of Linked Data principles into IoT applications and services.

This paper aims to explore these aspects, evaluating the role of LLMs in enhancing the TrUStAPIS methodology within the WoT framework and addressing the need for human oversight in ensuring the reliability and accuracy of LLM-generated outputs.

4 WoT-TrUStAPIS Enhancement

As we discussed earlier, in this paper we propose an enhancement of trust semantics in IoT implementing TrUStAPIS methodology that is aligned to WoT trust semantics, using LLMs.

First of all, we have to consider the fact that TrUStAPIS has been implemented as part of requirements phase in the trust framework presented in [9] that has been shown in Figure 1.

Thus, according to the main goal of the present work, the first step to be taken is to adapt the K-Model into the WoT SDLC [25] presented in Figure 2.

Analyzing the two models, it is possible to align the whole K-Model with the Manufactured phase of WoT in order to follow this methodology in the development of IoT entities. On the other hand, the utilization phase of the K-Model will be aligned to the four phases of the WoT SDLC: Bootstrapped, Operational, Maintenance and Destroyed.

Such alignment will benefit of the K-Model and TrUStAPIS methodology in the Manufactured Phase and will enhance and improve the utilization phase decomposing it in four phases. In fact, when a device has been completed and sold to the customers, it is fundamental to support it, updated it and eventually dispose it. However, in this paper we focus on the Requirements phase. Thus, in order to align TrUStAPIS methodology and its JSON template to the WoT JSON-LD schema, we need to add WoT elements to the conceptual model described and developed in [3].

In order to perform this task, we have decided to use firstly ChatGPT 3.5 and its "twin" ChatPDF [8]. We performed fine-tuning on the LLMs using prompt engineering in order to get them familiar with both TrUStAPIS methodology and WoT. We have decided to use ChatGPT 3.5 because it is free and available for everyone and ChatPDF because we could feed directly the LLM with the

[8] https://www.chatpdf.com/

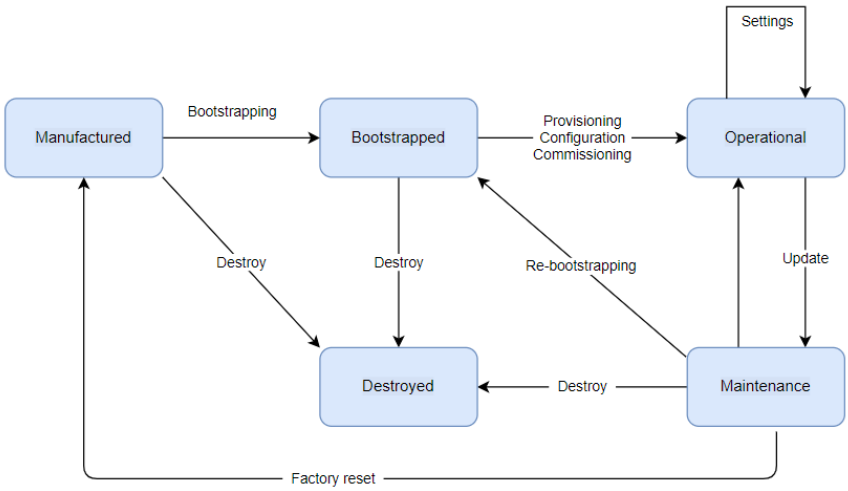


Fig. 2. WoT System Development Life Cycle [25].

TrUStAPIS PDF [3]. Then, Gemini has been used to enforce the new methodology and propose use case implementations.

Therefore, performing a fine tuning analysis with ChatGPT and ChatPDF, the LLMs suggested to implement two important elements belonging to WoT into the TruStAPIS methodology: events and ontologies. An event shall be connected to the elements goal and context. It is triggered by reaching the goal and it depends on the context. On the other hand, the element ontology is connected to the context and define common aspects related to an IoT device acting in a particular context (i.e., a smart home). All these elements are presented in the refined conceptual model depicted in Figure 3 (in grey and bold the new elements). Thus, after these preliminary tasks, we can now proceed introducing IoT trust semantics generated with the assistance of LLMs.

5 Prompts and Output Examples

In this experiment, we have started giving a role to ChatGPT as an assistant Requirements Engineer [9] and asked if it had knowledge about the IEEE 830-1993 specification. After its affirmative response, we have feed it with TrUStAPIS paper [3] in order to make the LLM aware of the basic features of the methodology.

After this task, basic questions about definition of the domains, characteristics and requirements analysis about TrUStAPIS have been provided to the LLM in order to understand if the methodology was correctly apprehended, in order to proceed and perform the modification explained in the previous section.

[9] https://www.technolynx.com/post/chatgpt-cheat-sheet

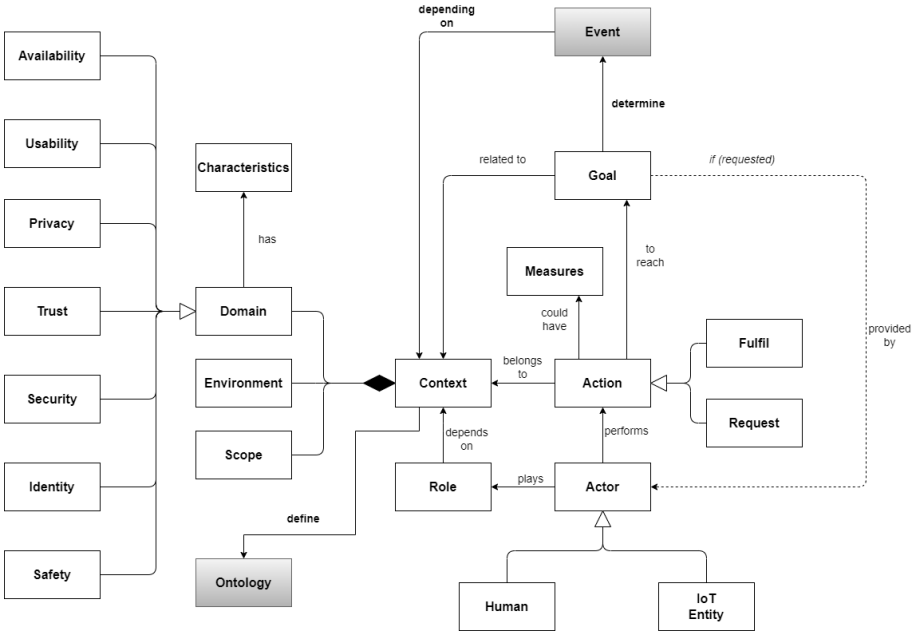


Fig. 3. TrUStAPIS Conceptual Model v2.0.

Finally, it was possible to pass to the following step related to IoT trust semantics generation to improve JSON-LD by creating ontologies in JSON syntax for the enhanced methodology.

5.1 IoT Trust Semantics for WoT-TrUStAPIS

In this phase, we have asked ChatGPT to include IoT trust semantics for TrUStAPIS methodology aligned to WoT. The LLM suggested five improvements: trust level, trust evaluation criteria, trust model, trustworthiness factors and trust policies. For space limitations, we do not provide the prompts here. However, they are available in the repository [10].

- **Trust Level**
  - Semantics Annotation: Define the concept of "Trust Level" in the ontology with properties such as "trustLevelValue" and "trustLevelDescription".
  - Linked Data: Link the Trust Level concept to relevant entities in the WoT ecosystem, such as IoT devices, services, or platforms, to indicate the level of trustworthiness associated with each entity.
- **Trust Evaluation Criteria**

[10] https://github.com/ferrarisUMA/ICICSpaper

  - Semantics Annotation: Specify trust evaluation criteria, such as "SecurityScore", "PrivacyScore", and "ReliabilityScore", within the ontology.
  - Linked Data: Establish relationships between Trust Evaluation Criteria and specific requirements in the TrUStAPIS JSON template to indicate how trustworthiness is assessed based on security, privacy, and reliability factors.
- **Trust Model**
  - Semantics Annotation: Define different trust models, such as "DirectTrustModel" and "ReputationTrustModel", with properties describing the characteristics of each model.
  - Linked Data: Link Trust Models to IoT entities and interactions to represent how trust relationships are established and maintained in the WoT environment.
- **Trustworthiness Factors**
  - Semantics Annotation: Capture trustworthiness factors like "DataIntegrity", "AuthenticationMechanism", and "ConsentManagement" in the ontology.
  - Linked Data: Associate Trustworthiness Factors with specific actions or goals in the TrUStAPIS JSON template to indicate the requirements related to ensuring trust in data integrity, authentication, and consent management.
- **Trust Policy**
  - Semantics Annotation: Define Trust Policies specifying rules and guidelines for establishing trust relationships in IoT systems.
  - Linked Data: ink Trust Policies to actors and contexts in the TrUStAPIS methodology to ensure that trust-related decisions and actions align with the defined policies and requirements.

We conjecture that by incorporating trust levels, evaluation criteria, trust models, trustworthiness factors, and trust policies into TrUStAPIS, we aim to enhance its ability to manage trust in IoT systems. Trust levels categorize entities by trust, evaluation criteria standardize assessments, a trust model provides a theoretical framework, trustworthiness factors quantify trust, and trust policies govern trust-related decisions. These additions improve decision-making, risk management, and collaboration, fostering security, reliability, and resilience in IoT environments.

5.2 JSON-LD

In order to implement JSON-encoded ontologies for TrUStAPIS enhancing them with the WoT elements specified earlier, we have started feeding ChatGPT with the original JSON template of TrUStAPIS [3]. Then, we have provided LLMs with a sample of JSON code related to an Identity Requirement shown in Figure 4.

In addition to the Identity Requirement encoded in JSON, we have provided also examples of trust and privacy requirements. For space limitation, we summarize here the important aspects related to them. The complete JSON code

```
"IoT_Requirement_IDNT01" : {
  "Context" : {
    "Domain" : [{
        "Identity" : {
         "Characteristic" : ["Authentication, Authorization"] } }],
    "Environment" : "Smart Home",
     "Scope" : "Entertainment" },
  "Actor" : {
     "Role" : ["Guest"],
      "Type" : ["Human"]},
  "Action" : {
     "Type" : ["Request"]
     "Measure" : [" "]},
  "Goal" : "In order to be trusted, a user must be identified"
}
```

Fig. 4. Identity Requirement in TrUStAPIS v1.0 using JSON encoding.

can be found in [3]. For the trust requirement, we consider the characteristics *direct* and *local*. The environment and the scope were the same for the Identity Requirement (even for the Privacy Requirement). The roles are *owner* or *guest* and the type of the actor is *human*. The type of the action is *request* and the goal is: "*A trusted user can listen to the friend's music by speakers or earphones*". On the other hand, for the privacy requirement we consider *confidentiality* as a characteristic. The role of the actor is only the *owner* of the *smart-home*. Then, the action type is *request* and the goal is "*Outside the trusted circle, nobody can know which music is listened by the owner*".

Thus, we have asked ChatGPT to provide an example of JSON code related to the security domain. The output is presented in Figure 5.

During the prompt engineering, details related to the characteristics of Security described in [3] have not been provided to the LLM. However, its output contained two characteristics related to security: Encryption and Access Control. In [3], the security characteristics are: Authentication, Authorization, Integrity, Confidentiality, Delegation and Non-repudiation. We can observe that Access Control can be a conjunction of Authentication and Authorization. Encryption can be connected to integrity and confidentiality.

On the contrary, for the type of action, without prompting details of them, the LLM provides a set different from ours (i.e., configure). Then, it provides a goal and filled the JSON in a useful way. With few inputs, the LLM could understand how to construct a JSON code related to security domain.

Therefore, we asked to ChatGPT to provide outputs related to three other domains of TrUStAPIS: Safety, Availability and Usability. For these cases, the domain characteristics outputs are the following. For safety it suggests "Emergency

```
"IoT_Requirement_SECU01" : {
    "Context" : {
        "Domain" : [{
            "Security" : {
                "Characteristic" : ["Encryption, Access Control"]
            }
        }],
        "Environment" : "Smart Home",
        "Scope" : "Entertainment"
    },
    "Actor" : {
        "Role" : ["Administrator"],
        "Type" : ["Human"]
    },
    "Action" : {
        "Type" : ["Configure"],
        "Measure" : ["Use strong encryption for communication"]
    },
    "Goal" : "Ensure secure communication and access control in the smart home system"
}
```


Fig. 5. ChatGPT output for the Security Requirement of TrUStAPIS v1.0, encoded in JSON.

Shutdown" and "Fault Tolerance". In TrUStAPIS the safety characteristics are: Feedback, Protection, Resilience and Integrity. We can considere that the two proposed characteristics can be covered by Protection and Resilience. With respect to Availability, ChatGPT proposed Redundancy and Fault Recovery. In TrUStAPIS, we have: Resilience, Scalability, Redundancy and Integrity. We can observe that one characteristic is the same (without specifying it) and the second can be referred to resilience. It is important to underline that the same characteristic can be present in different domains but their difference reflect the difference of the domains [3]. Finally, with respect to Usability, the suggested characteristics are: "Intuitive Interface" and "User Feedback". In TrUStAPIS: Effectiveness, Efficiency, Simplicity, Understandability, Accessibility, Flexibility and Reliability. We can observe that Understandability can be linked to the first suggested characteristic and the second one cannot be related to any of those.

After the output provided by ChatGPT, we moved forward asking it to adapt the JSON codes of TrUStAPIS to the WoT model using JSON-LD encoding. An example of one of the outputs is depicted in Figure 6, where we can see that the JSON code has been adapted to the JSON-LD style. At the top of the code there are all the defined parameters for TrUStAPIS collected as domains.

Some consideration that we can make are related to the context and to the ID. The context has became central and it is connected to all the parameters involving the TrUStAPIS conceptual model presented and Requirements definition. This is an important difference and it can be considered a valid one because the context is crucial for IoT. According to it, everything else can be performed and considered differently. However, in TrUStAPIS methodology the other parameters were bounded to context reflecting in a similar way this aspect. On the other, hand ID has been extended. In fact, in TrUStAPIS methodology each requirements have their ID reflecting the domain and their number. Such requirements, must be stored in a separate database in order to be available

```
{
  "@context": {
    "domain": "http://example.org/vocab#",
    "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "Trust": "domain:Trust",
    "Characteristic": "domain:Characteristic",
    "Context": "rdf:Description",
    "Environment": "domain:Environment",
    "Scope": "domain:Scope",
    "Actor": "domain:Actor",
    "Role": "domain:Role",
    "Type": "domain:Type",
    "Action": "domain:Action",
    "Measure": "domain:Measure",
    "Goal": "domain:Goal",
    "Event": "domain:Event"
  },
  "@id": "http://example.org/requirements/IoT_Requirement_TRST01",
  "@type": "Context",
  "domain:Domain": {
    "@type": "Trust",
    "domain:Characteristic": ["Direct", "Local"]
  },
  "domain:Environment": "Smart Home",
  "domain:Scope": "Entertainment",
  "domain:Actor": {
    "@type": "Actor",
    "domain:Role": ["Owner", "Guest"],
    "domain:Type": ["Human"]
  },
  "domain:Action": {
    "@type": "Action",
    "domain:Type": ["Request"],
    "domain:Measure": [""]
  },
  "domain:Goal": "A trusted user can listen to the same music as a friend by speakers or earphones",
  "domain:Event": "Music sharing request"
}
```


Fig. 6. ChatGPT output for the Trust Requirement of TrUStAPIS v2.0, encoded in JSON-LD.

and this will enhance an important characteristic of TrUStAPIS methodology: traceability.

Furthermore, it can be observed that the new conceptual model is fullfilled because it contains the element "domain:Event" and it is has been assigned the value "Music sharing request" that is precisely an event that we can expect according to the Goal.

As we have mentioned earlier, traceability is a core part of TrUStAPIS. Thus, we have asked ChatGPT to suggest which requirements can be connected between them by traceability considering the one discussed. The output has been the one presented in Figure 7. We can see that ChatGPT considered traceability according to the possible contexts (if shared or if there is the same actor involved). However, this output has not been completely satisfying because traceability can be considered also for other parameters (i.e., goals and contexts). Considering the fact that traceability is an important feature, in this case, even if LLMs can process a lot of data, a human intervention is still fundamental.

For a complete description of the characteristics and traceability concepts, please refer to [3].

6 Experiments Output Analysis

In this phase, the outputs provided in the previous section by ChatGPT and ChatPDF, and modified by human supervision, have been also evaluated with Gemini LLM.. After fine-tuning, we have asked the LLM to provide three different use cases. However, in this paper we discuss only the first one and its

- **Identity Requirement (IoT_Requirement_IDNT01) and Usability Requirement (IoT_Requirement_USAB01):**
  - Shared Context: Smart Home, Entertainment
  - Shared Actor: User (Human)
- **Trust Requirement (IoT_Requirement_TRST01) and Security Requirement (IoT_Requirement_SECU01):**
  - Shared Context: Smart Home, Entertainment
- **Privacy Requirement (IoT_Requirement_PRIV01) and Usability Requirement (IoT_Requirement_USAB01):**
  - Shared Context: Smart Home, Entertainment
  - Shared Actor: Owner (Human)
- **Availability Requirement (IoT_Requirement_AVAIL01) and Usability Requirement (IoT_Requirement_USAB01):**
  - Shared Context: Smart Home, Entertainment
  - Shared Actor: User (Human)

Fig. 7. Traceability among requirements suggested by ChatGPT.

specializations, the others are available in the GitHub repository (see previous footnote).

6.1 Use Case: Smart Thermostat in Smart Home (Context-aware Trust)

- TrUStAPIS Semantics:
  - Domain: Data Integrity (Temperature Accuracy)
  - Indicator: Tolerance level for temperature readings
  - Trust Level: Based on LLM analysis, user comfort and energy efficiency are key concerns. Thus, the trust level is set to +/- 0.5 degrees Celsius.
  - Action: Users can set target temperature through the WoT TD interaction.
- WoT Thing Description (TD) Mapping:
  - Property: "targetTemperature" (annotated with "dataIntegrity")
  - Sub-property: "tolerance" (set to 0.5 degrees Celsius)
  - Interaction: "setTargetTemperature" (with output parameter "currentTemperature" annotated with "dataIntegrity")

As we can see above, the first output related to the use case considers few elements. However, for TrUStAPIS, we can see that it contains an important aspect such as a domain that is a composition of security and availability. The metric proposed for it is related to tolerance levels. Then, we can see the IoT trust semantic "Trust Level" where different parameters are considered such as the user comfort against energy efficiency, and a justification is provided by the LLMs. Moreover, as we have proposed a new WoT-TrUStAPIS methodology, the

LLM provided an output according to it considering important aspects of the proposed use case. However, such use case is limited, so we have asked the LLM to provide a more elaborated output.

6.2 Use Case Specialization

In order to design and evaluate a specialized scenario, we have asked Gemini to improve the use case focusing especially on the new elements provided by the improvement of TrUStAPIS methodology according to the WoT elements (i.e., events and ontologies) considering also context (for different domains), actors and different goals.

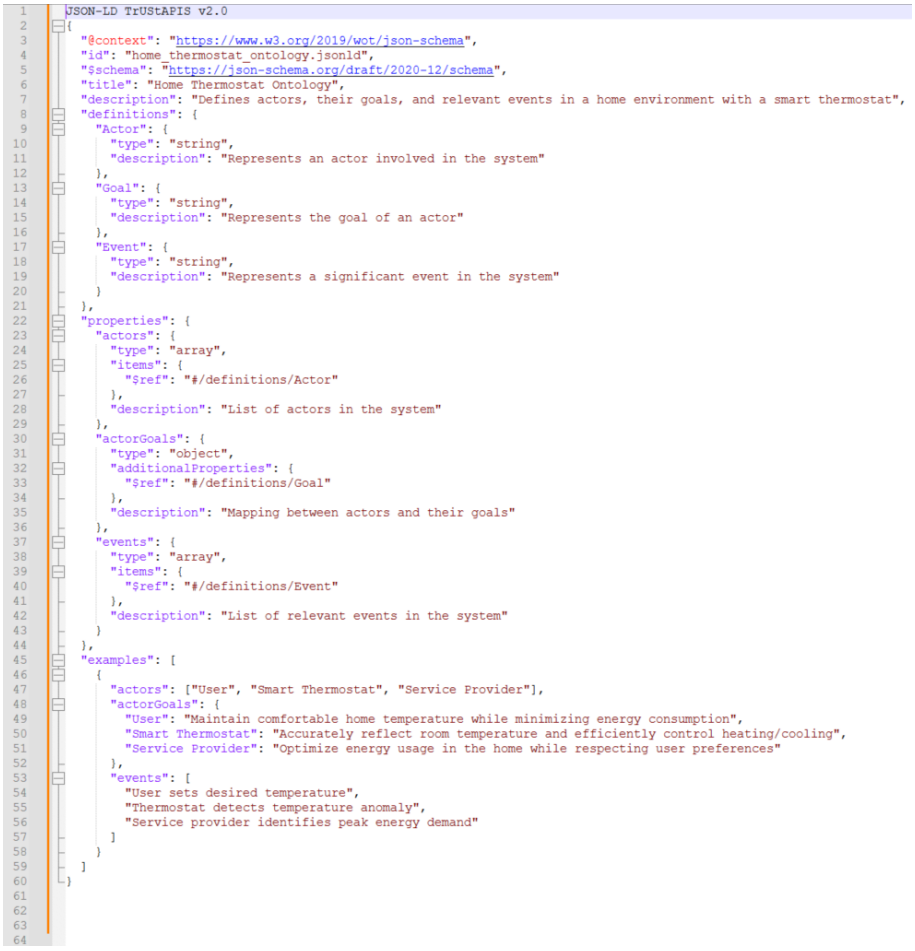
The example prompt output, provided earlier in the GitHub link, is the following.

- Smart Thermostat in a Home (Context-aware Trust with Multiple Actors)
  - Context: A smart thermostat (Thing) in a home environment with multiple residents (Users) and a remote energy management service (Service Provider).
  - Actors:
    - Users: Home residents with varying preferences for comfort and energy efficiency.
    - Smart Thermostat (Thing): Manages heating and cooling based on user settings and environmental conditions.
    - Service Provider: Offers remote energy management services (e.g., optimizing energy usage during peak hours).
  - Goals:
    - Users: Maintain comfortable home temperature while minimizing energy consumption.
    - Thermostat: Accurately reflect room temperature and efficiently control heating/cooling.
    - Service Provider: Optimize energy usage in the home while respecting user preferences.
  - Events:
    - User sets desired temperature through the WoT TD interaction (action: "setTargetTemperature").
    - Thermostat detects a significant temperature deviation from the set point (event: "temperatureAnomalyDetected").
    - Service Provider identifies peak energy demand period (event: "peakEnergyDemand").
  - Trust Domains and Considerations:
    - Data Integrity (Temperature Accuracy):
      - Users: Trust the thermostat to accurately reflect room temperature for optimal comfort.
      - LLM analysis considers user reviews emphasizing comfort and energy efficiency.
      - TrUStAPIS defines a +/- 0.5 degrees Celsius tolerance level.

    - WoT TD maps this to a "tolerance" sub-property within the "targetTemperature" property.
* Data Privacy:
    - Users: Trust that temperature data is collected, stored, and used only for controlling the thermostat or with their consent.
    - Service Provider needs some temperature data to optimize energy usage but should respect user privacy preferences.
    - LLM analysis considers regulations on data privacy and user concerns about data collection.
    - TrUStAPIS defines user consent management for data sharing with the Service Provider.
    - WoT TD interaction for sharing data with the service provider requires explicit user consent.
    - The TD property "collectedTemperatureData" is annotated with "privacy" and linked to the service provider's privacy policy.
* Availability:
    - Users: Expect the thermostat to be functional and responsive to their commands.
    - Service Provider needs access to the thermostat for optimization purposes, but shouldn't disrupt user control.
    - LLM analysis considers user reviews on responsiveness and potential service disruptions.
    - TrUStAPIS defines a minimum uptime requirement for the thermostat.
    - WoT TD interaction for service provider access might be restricted to specific times or require user confirmation.
* Additional Considerations:
    - Context-aware Adjustments: The thermostat can adjust its operation based on additional context (e.g., occupancy sensors, weather data).
    - Dynamic Trust Levels: User preferences and context might influence the acceptable tolerance level for temperature accuracy (e.g., tighter range at night for sleep).
    - Explainability: LLM-generated justifications can inform users about the rationale behind trust requirements and data usage practices.

This example demonstrates how trust management becomes more complex with multiple actors and conflicting goals. However, only through more complex scenarios is possible to analyze the different possibilities provided by the methodology, and applying TrUStAPIS is possible to elicit the proper requirements. In this case, the context is properly explained, the actors are the one important for such use case and the goals and events along with the domains enable developers to understand if conflicts arise and how to solve them. A possibility to solve such conflicts has been presented in [26].

Finally, we have asked the LLM to generate WoT semantics related to the use case using JSON-LD. The output is presented in Figure 8.



```
JSON-LD TrUStAPIS v2.0
{
  "@context": "https://www.w3.org/2019/wot/json-schema",
  "id": "home_thermostat_ontology.jsonld",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "Home Thermostat Ontology",
  "description": "Defines actors, their goals, and relevant events in a home environment with a smart thermostat",
  "definitions": {
    "Actor": {
      "type": "string",
      "description": "Represents an actor involved in the system"
    },
    "Goal": {
      "type": "string",
      "description": "Represents the goal of an actor"
    },
    "Event": {
      "type": "string",
      "description": "Represents a significant event in the system"
    }
  },
  "properties": {
    "actors": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/Actor"
      },
      "description": "List of actors in the system"
    },
    "actorGoals": {
      "type": "object",
      "additionalProperties": {
        "$ref": "#/definitions/Goal"
      },
      "description": "Mapping between actors and their goals"
    },
    "events": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/Event"
      },
      "description": "List of relevant events in the system"
    }
  },
  "examples": [
    {
      "actors": ["User", "Smart Thermostat", "Service Provider"],
      "actorGoals": {
        "User": "Maintain comfortable home temperature while minimizing energy consumption",
        "Smart Thermostat": "Accurately reflect room temperature and efficiently control heating/cooling",
        "Service Provider": "Optimize energy usage in the home while respecting user preferences"
      },
      "events": [
        "User sets desired temperature",
        "Thermostat detects temperature anomaly",
        "Service provider identifies peak energy demand"
      ]
    }
  ]
}
```

Fig. 8. JSON-LD TrUStAPIS v2.0 description related to the use case.

We see that the elements of the JSON template have been expanded in order to be aligned to the WoT. However, Gemini's output needed refinements in order to obtain the final output presented in Figure 8.

In conclusion, we can state that LLMs are of paramount importance for the development of IoT entities considering different domains such as privacy, security or trust. However, human involvement (i.e., human-machine teaming) is still fundamental.

7 Conclusions and Future Work

In this paper, we have been using the potentialities offered by LLMs in order to extend and revise with WoT trust semantics the TrUStAPIS methodology. We

have used two different LLMs in order to perform and evaluate different tasks. We have used ChatGPT version 3.5 in order to perform an enhancement of TrUStAPIS methodology considering WoT elements. Then, we asked the LLM to define JSON encodings related to the methodology by upgrading it with the JSON-LD format used by WoT. We believe that such improvement can benefit TrUStAPIS methodology in order to be considered and implemented in IoT devices Development Lifecycle covering the Requirements Engineering phase. We have then proposed a summarization of the outputs generated by the LLM, if they were expected or not, along with their feasibility. Finally, we have used Gemini to generate distinct use cases to further validate the output provided by ChatGPT. It must be stated that LLMs outputs have been really assistive in both tasks, however human intervention have been necessary.

In future work, we plan to proceed with the improvement of TrUStAPIS methodology, focusing more on traceablity and domains analysis. We will also further explain and integrate in an extended version of this paper useful information provided now only on Github. Moreover, we will consider other LLMs providing a comparison of their outputs in terms of accuracy and time spent in providing a more accurate and complete answer.

Acknowledgments

This work has been partially supported by the SECAI project funded by the Spanish Ministry of Science and Innovation and the Research State Agency (PID2022-139268OB-I00).

References

1. Roman, R., Najera, P., & Lopez, J. (2011). Securing the internet of things. Computer, 44(9), 51-58.
2. Gupta, B. B., & Quamara, M. (2020). An overview of Internet of Things (IoT): Architectural aspects, challenges, and protocols. Concurrency and Computation: Practice and Experience, 32(21), e4946.
3. Ferraris, D., & Fernandez-Gago, C. (2020). TrUStAPIS: a trust requirements elicitation method for IoT. International Journal of Information Security, 19(1), 111-127.
4. Konsta, A. M., Lafuente, A. L., & Dragoni, N. (2023). A Survey of Trust Management for Internet of Things. IEEE Access.
5. Jøsang, A., Ismail, R., & Boyd, C. (2007). A survey of trust and reputation systems for online service provision. Decision support systems, 43(2), 618-644.
6. McKnight, D. H., & Chervany, N. L. (2000). What is trust? A conceptual analysis and an interdisciplinary model.
7. Hoffman, L. J., Lawson-Jenkins, K., & Blum, J. (2006). Trust beyond security: an expanded trust model. Communications of the ACM, 49(7), 94-101.
8. Pavlidis, M. (2011, December). Designing for Trust. In CAiSE (doctoral consortium) (pp. 3-14).

9. Ferraris, D., Fernandez-Gago, C., & Lopez, J. (2018, February). A trust-by-design framework for the internet of things. In 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS) (pp. 1-4). IEEE.
10. Yu, E. (2010). Modeling strategic relationships for process reengineering.
11. Massacci, F., Mylopoulos, J., & Zannone, N. (2010). Security requirements engineering: the SI* modeling language and the secure tropos methodology. In Advances in Intelligent Information Systems (pp. 147-174). Berlin, Heidelberg: Springer Berlin Heidelberg.
12. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., & Mylopoulos, J. (2004). Tropos: An agent-oriented software development methodology. Autonomous Agents and Multi-Agent Systems, 8, 203-236.
13. Mouratidis, H., & Giorgini, P. (2007). Secure tropos: a security-oriented extension of the tropos methodology. International Journal of Software Engineering and Knowledge Engineering, 17(02), 285-309.
14. Rios, R., Fernandez-Gago, C., & Lopez, J. (2018). Modelling privacy-aware trust negotiations. Computers & Security, 77, 773-789.
15. Mavropoulos, O., Mouratidis, H., Fish, A., Panaousis, E., & Kalloniatis, C. (2016). Apparatus: Reasoning about security requirements in the internet of things. In Advanced Information Systems Engineering Workshops: CAiSE 2016 International Workshops, Ljubljana, Slovenia, June 13-17, 2016, Proceedings 28 (pp. 219-230). Springer International Publishing.
16. IEEE Computer Society. Software Engineering Standards Committee, & IEEE-SA Standards Board. (1998). IEEE recommended practice for software requirements specifications (Vol. 830, No. 1998). IEEE.
17. Sporny, M., Longley, D., Kellogg, G., Lanthaler, M., & Lindström, N. (2020). JSON-LD 1.1. W3C Recommendation, Jul.
18. Wood, D., Zaidman, M., Ruth, L., & Hausenblas, M. (2014). Linked Data. Manning publications co.
19. Jara, A. J., Olivieri, A. C., Bocchi, Y., Jung, M., Kastner, W., & Skarmeta, A. F. (2014). Semantic web of things: an analysis of the application semantics for the iot moving towards the iot convergence. International Journal of Web and Grid Services, 10(2-3), 244-272.
20. Elsayed, K. I., & Elgamel, M. S. (2020, December). Web of things interoperability using JSON-LD. In 2020 30th International Conference on Computer Theory and Applications (ICCTA) (pp. 31-35). IEEE.
21. Alivanistos, D., Santamaría, S. B., Cochez, M., Kalo, J. C., van Krieken, E., & Thanapalasingam, T. (2022). Prompting as probing: Using language models for knowledge base construction. arXiv preprint arXiv:2208.11057.
22. Li, T., Huang, W., Papasarantopoulos, N., Vougiouklis, P., & Pan, J. Z. (2022). Task-specific pre-training and prompt decomposition for knowledge graph population with language models. arXiv preprint arXiv:2208.12539.
23. Pitis, S., Zhang, M. R., Wang, A., & Ba, J. (2023). Boosted prompt ensembles for large language models. arXiv preprint arXiv:2304.05970.
24. Jiang, Z., Xu, F. F., Araki, J., & Neubig, G. (2020). How can we know what language models know?. Transactions of the Association for Computational Linguistics, 8, 423-438.
25. https://www.w3.org/TR/wot-architecture
26. Ferraris, D., Fernandez-Gago, C., & Lopez, J. (2023). POM: a trust-based ahp-like methodology to solve conflict requirements for the IoT. In Collaborative Approaches for Cyber Security in Cyber-Physical Systems (pp. 145-170). Cham: Springer International Publishing.