Further with Knowledge Graphs. M. Alam et al. (Eds.)
AKA Verlag and IOS Press, 2021

doi:10.3233/SSW210031

TODO: A Core Ontology for Task-Oriented Dialogue Systems in Industry 4.0

Cristina ACETA [a,1], Izaskun FERNÁNDEZ [a] and Aitor SOROA [b]
[a] *TEKNIKER, Basque Research and Technology Alliance (BRTA), Spain*
[b] *CCIA Group, University of the Basque Country, Spain*

Abstract. Nowadays, the demand in industry of dialogue systems to be able to naturally communicate with industrial systems is increasing, as they allow to enhance productivity and security in these scenarios. However, adapting these systems to different use cases is a costly process, due to the complexity of the scenarios and the lack of available data. This work presents the Task-Oriented Dialogue management Ontology (TODO), which aims to provide a core and complete base for semantic-based task-oriented dialogue systems in the context of industrial scenarios in terms of, on the one hand, domain and dialogue modelling and, on the other hand, dialogue management and tracing support. Furthermore, its modular structure, besides grouping specific knowledge in independent components, allows to easily extend each of the modules, attending the necessities of the different use cases. These characteristics allow an easy adaptation of the ontology to different use cases, with a considerable reduction of time and costs. So as to demonstrate the capabilities of the the ontology by integrating it in a task-oriented dialogue system, TODO has been validated in real-world use cases. Finally, an evaluation is also presented, covering different relevant aspects of the ontology.

Keywords. Semantic Web, Dialogue Systems, Natural Language Processing, Industry 4.0.

1. Introduction

Factory workers are a core factor in production environments. These environments are becoming more automatized over time, and workers require of intuitive and powerful interaction techniques so as to successfully perform their assigned tasks in collaboration with automatisms [1]. To cover this necessity, Human-Machine Interfaces (HMI) have increasingly evolved in last years with the development of new mobile techniques and new gadgets such as smartphones, tablets or Augmented Reality (AR) glasses. In this context, a big number of systems have been developed, especially in collaborative robotics, with human-machine interaction capabilities in different degrees [2,3].

In this sense, task-oriented dialogue systems are a very useful tool that allow workers to work on multiple tasks at once without reducing the quality of their work, by perform-

[1]Corresponding Author: Cristina Aceta, Tekniker, C/Iñaki Goenaga 5, 20600 Eibar, Spain; E-mail: cristina.aceta@tekniker.es.

ing secondary tasks simply by communicating with the target system. It is also worth noting that the capacity of dialogue systems of communicating in natural language has a positive impact in acceptation from humans [4]. However, the capacity of the system to interpret human commands in most current frameworks in industrial environments is basically based on a predefined vocabulary that allows the worker to interact with the target system, making use of templates [5,6]. These templates are based on human-readable models, but all of them depend on expert manual work, which in most cases supposes high costs, and they cannot be directly reused in other scenarios. Furthermore, these approaches negatively affect the naturality of the interaction and, thus, human acceptation [7]. Also, the capacity of current interfaces to semantically understand natural human commands is still limited and the required effort to do so is usually high.

This evidence motivates the development of the ontology presented in this work: the Task-Oriented Dialogue management Ontology (TODO). TODO is a core, modular ontology that provides task-oriented dialogue systems with the necessary means to be capable of naturally interacting with workers (both at understanding and at communication level) and that can be easily adapted to different industrial scenarios, reducing adaptation time and costs. Moreover, it allows to store and reproduce the dialogue process to be able to learn from new interactions. To the best of the authors' knowledge at the time of presenting this work, there are not core ontologies in the literature that deal with natural interaction in industrial scenarios at this level, which gives special relevance to TODO.

This paper is organized as follows: Section 2 provides related work that is relevant to this paper. Section 3 presents TODO, following the methodology used to develop it; Section 4 makes some remarks on the instantiation process of TODO and its use in several industry-related use cases. Finally, Section 5 includes a set of final considerations for this work.

2. Related Work

According to [8], the basic architecture of a task-oriented dialogue system consists of a **natural language understanding** component, a **dialogue state tracker**, a **dialogue policy** and a **natural language generation** module. The first aims to extract an interpretation from the command, the second and the third ones deal with the dialogue process and management, and the latter generates the response directed to the user.

In general, for the natural language understanding component, most modern task-oriented dialogue systems are somewhat based in frames, which consist of a representation on the information to be provided as slots, to be filled with the information provided by the user [8] and supported by a knowledge base [9]. To assure the correct interpretation for a given command in this kind of dialogue systems, natural language technologies are used in several solutions in the literature [10,11]. Classical architectures made use of rules to detect the intent of the user and to perform slot filling, mainly semantic grammars, as it can be seen in [9], or templates [12]. However, modern approaches generally do not make use of rules and rely on machine-learning-based (both classic machine learning and deep learning) techniques [8]. Nevertheless, and as pointed out by [8], industrial approaches often make use of rules and templates for slot-filling techniques, as the domain is limited enough for this approach to work.

As for dialogue management, rules have also been traditionally used in task-oriented dialogue systems [13]. However, and as in the previous case, these techniques are being

replaced by machine-learning-based methods such as conditional random fields [14,15], maximum entropy models [16] and, more recently, deep learning models [17,18] [19].

Although they are proven to be widely used, machine-learning-based methods require of great amounts of data to train the systems, which is not easy to obtain for industrial task-oriented dialogue systems, and rules are often generated for both natural language understanding and dialogue management. However, constructing rules is time and cost consuming and may be prone to errors, and supervised machine learning techniques are being added to the paradigm by combining them with these rules to optimize results [20,21].

In current approaches, ontologies have also been considered both for the natural language understanding and the dialogue management components of task-oriented dialogue systems, as they are a powerful tool that allows to define in detail the domain and reduce ambiguity between agents [10]. However, most dialogue systems that use ontologies found in the literature are limited to highly specific use cases and mainly to model the domain. In [22], for the banking and finance domain, domain information, such as products and services, is modelled in the ontology, as well as certain state-related dialogue information (e.g., which is the current product that is discussed in the conversation). In industrial scenarios, the work in [11] makes use of an ontology to model the domain in terms of possible actions to be performed by the robot and a description of the scenario.

As far as the authors of this paper are concerned, OntoVPA [23] is currently the only intent to achieve a generic approach to semantic-based task-oriented dialogue systems both at domain and dialogue management level. This commercial tool, aims to provide a general approach to manage dialogue through ontologies and, more precisely, by making a distinction between a domain and a dialogue management ontology. The domain ontology and its instantiation store the knowledge related to the domain and the slots to be filled for any of the modelled actions. The dialogue ontology, which is inspired by the Speech Act Theory [24], has the capability of managing the dialogue process, perform state tracking and can also manage responses and answers. However,the documentation for this tool is limited and the ontologies developed are not publicly available, what makes its reuse impossible.

The advances in last decades regarding ontologies have allowed to associate their building process to an engineering task through the creation of methodologies for that matter to obtain quality ontologies [25]. However, as pointed by [25], traditional methodologies such as METHONTOLOGY [26] or NeOn [27] propose actions that are time and resource consuming. Modern methodologies –such as eXtreme Method [28] or RapidOWL [29]–, on the other hand, tend to include guidelines that are more consumption friendly, but do not consider basic characteristics of Linked Data such as reuse of ontologies or ontology maintenance and updating [25]. For that purpose, [30] have defined Linked Open Terms (LOT), a light methodology that also complies with reuse and maintenance aspects. Moreover, this methodology has an industrial version, especially optimised for ontology building for industrial scenarios.

The remarks above illustrate that the use of ontologies is a current trend in the literature referring to task-oriented dialogue systems, especially for *ad-hoc* domain modelling, with great capabilites, but still in an early stage for dialogue management and generic implementations. Indeed, since the only approach for this last issue is not available to reuse, a new ontology development is motivated.

3. The Task-Oriented Dialogue management Ontology (TODO)

With the aim of enhancing natural communication between workers in industrial environments and the systems to be used by them, TODO (Task-Oriented Dialogue management Ontology) has been developed to be the core of task-oriented dialogue systems. This section describes TODO in detail, following the methodology used for that purpose.

3.1. Ontology Development Methodology

In ontology development, two main considerations arise: on the one hand, ontologies have to be "carefully designed and implemented" [31], so as to properly model all the necessary information for their final use. On the other hand, ontology development is becoming more and more centered in reuse [32]. Considering the above, it is important to follow a well-defined design methodology to develop ontologies that are optimal both for their intended function and to be reused by others. In the development process of TODO, the methodology followed is LOT (Linked Open Terms), in its industrial version [30], as it focuses on design of ontologies oriented to industrial scenarios. This methodology sets four main steps of development:

- **Requirements specification**. It defines the motivation and the requirements to be fulfilled by the ontology, through the Ontology Requirement Specification Document (OSRD) [33]. The ORSD defines the purpose, scope, intended uses and requirements –defined as Competency Questions (CQ)– of the ontology.
- **Implementation**. By considering the requirements set in the previous step, the ontology is constructed and evaluated.
- **Publication**. Once the ontology has been created and properly annotated, its documentation is generated, and both ontology and documentation are published and made accessible online.
- **Maintenance**. This step includes periodical revisions with the aim to solve issues, add improvements, etc.

The following sections will document TODO ontology design process in terms of the first three steps defined in LOT, as the last one is understood as further periodic maintenance work after an initial version of TODO has been released.

3.2. Requirements Specification Step

Besides the motivation of the ontology, which has been previously documented in the introduction, the ORSD leads to determine the specifications for the functional requirements of the ontology; that is, the knowledge that the ontology must cover. In this sense, it is important to bear in mind that the main objective of the dialogue is to obtain a command that is understandable for the target system from a natural language request.

To determine the required knowledge, three experts in collaborative industrial work and dialogue systems were interviewed to gather information about their necessities and the characteristics they considered to be covered by a task-oriented dialogue system and which type of interactions were expected. By using the information obtained, a series of requirements that had to be covered by the ontology were identified and codified as CQs. For TODO, a total of 93 CQs were obtained, which can be grouped into the following 10 basic CQs:

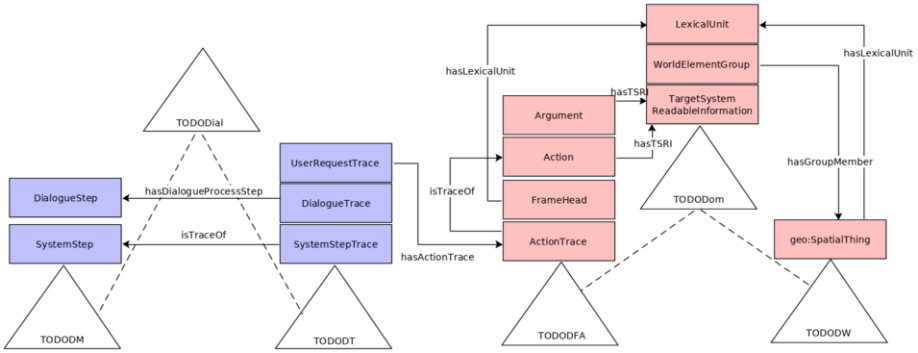

Figure 1. Overview of TODO. Triangles represent ontology modules, whereas rectangles correspond to classes.

- **CQ01**. What are the elements that are present in the scenario?
- **CQ02**. Which is the action to be performed by the target system given a series of key elements obtained from a user request?
- **CQ03**. Which are the arguments of a specific action?
- **CQ04**. Given a set of arguments from a specific action, to what argument can a key element from the user request be associated to?
- **CQ05**. Which is the format of the information that has to be provided to the target system?
- **CQ06**. Which is the first/next step of the dialogue?
- **CQ07**. What should be told to the user given a specific situation?
- **CQ08**. Given some output to the user, it is some input from the user required or not?
- **CQ09**. Which step is currently being performed in the dialogue?
- **CQ10**. Which is the trace –and the information that includes– for an element?

These CQs have helped to define the scope of the ontology and to delimit its different areas of knowledge, along with the classes and relations to be modelled. These CQs have also helped to define the criteria to search for relevant ontological resources for reuse, which will be specified in the implementation section.

3.3. Implementation Step

Ontology Implementation

TODO has been implemented as a modular ontology, inspired in the modules in [23], which distinguish between domain-related and dialogue-related information. This modular approach provides many beneficial aspects to ontologies in terms of maintenance, reasoning-processing[2], validation, comprehension, collaborative effort and reuse [34].

Considering the CQs, the domain and dialogue modules have been divided in sub-modules in TODO, so as to cover more specific areas of knowledge that have certain independence. The definition of the CQs also contributes to the identification of reusable concepts from other ontologies, a task that is considered as good practice in ontology

[2]Reasoners and processing tools may take more time to work over big ontologies.

development [35]. More details on the reused concepts for each module are specified in the following sections.

The general overview of TODO and its modules can be seen in Figure 1. The following lines will describe the dialogue- and domain-related modules, along with each of their submodules.

Dialogue Ontology Module - TODODial

The TODODial module, which covers basic CQs 06 to 10, deals with the concepts used for dialogue modelling, both for dialogue management and dialogue tracing. The knowledge covering the former is intended to be implemented as static, and the latter as dynamic, as it will be instantiated during the dialogue process. Considering this, the module consists of two submodules: **Dialogue Management** (TODODM) and **Dialogue Tracing** (TODODT).

TODODM, which covers basic CQs 06 to 08, models the concepts that allow to manage the dialogue process. In this sense, the dialogue process consists of two main types of steps: **Dialogue steps** and **Process steps**. The former require of some interaction with the user by the system (`SystemStep`) and, depending on the modelling of the dialogue step, it may prompt the user to obtain information (`SystemRequest`) or it may output some information (`SystemResponse`). The output to be presented to the user is defined in a data property (`outputSentence`).

A system response will typically imply a system request, although it may also imply a predefined action in dialogue control (`DialogueControlMarkers`), which might establish that the dialogue process needs to **continue**, **finish** or to **restart**.

On the other hand, process steps do not require from interaction with the user. Both **dialogue** and **process steps** have a **step function** associated. **Step functions** are a very important concept in the dialogue process, since they are directly linked to specific functions in the dialogue system component that manages the dialogue process (usually known as dialogue manager). Furthermore, **step functions** have a set of implications (through the object property `implies` and its subproperties) that determine the next step of the dialogue process, considering the output of the **step function**s when executed in the dialogue manager.

The **TODODT** module deals with basic CQs 09 to 10, and aims to model the necessary concepts to allow dialogue recreability; that is, to gather the necessary information to be able to reproduce the dialogue process, so as to detect possible errors or check the flow of a specific piece of dialogue. Furthermore, this information can be exploited to learn from finished interactions.

In this module, three main dialogue elements are subject to tracing: **dialogues**, **user steps** and **system steps**. For dialogues, two types are distinguished: **Dialogues** and **Secondary dialogues**. **Dialogues** are the main concept of the dialogue, whereas **Secondary dialogues** are the subdialogues that emerge in the context of a **Dialogue**. For example, given the user request "I want information": a dialogue trace would be created, and that dialogue trace would have an associated user request trace. However, the user has not specified the element they want information about, and the system generates a secondary dialogue, with a trace of the request that asks for the missing information –which will

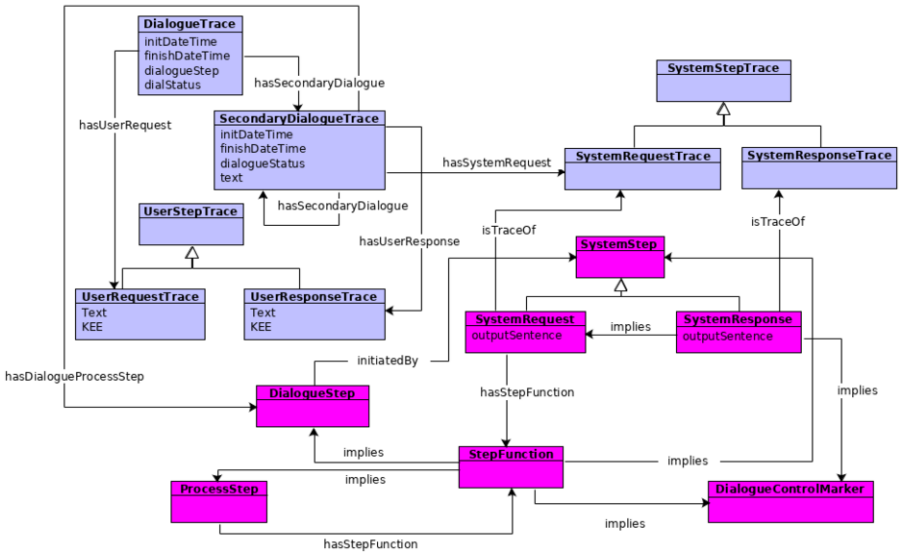

Figure 2. Simple overview of TODODial.

recreate the output sentence(s) that the user has received as output. If the system output is a request, a trace for the user response will also be created.

Figure 2 shows a simple overview of the main relations in TODODial, showing the Tracing module in purple and the Management module in magenta. As it can be seen, both modules are joined in terms of traces.

Domain Ontology Module - TODODom

The TODODom module, which covers basic CQs 01 to 05, deals with the concepts used for domain modelling, in terms of the actions that the target system can perform and the elements that are present in the scenario. Most importantly, this module is in charge of obtaining a target-system-readable command from a natural interaction.

Taking into account these considerations, this module includes two submodules: **Frame-Action** (TODODFA), which deals with the former, and **World** (TODODW), which deals with the latter.

TODODFA, which covers basic CQs 02 to 05, models the actions that can be performed by the target system and the arguments required by said action to be executed. It also establishes the concepts that help identify the action from a natural command.

This module, which is inspired by the GUS architecture [36], considers the **skills** of the target system[3], and associates each **skill** to an **intent**, which is the user objective when directing a request to the dialogue system (e.g. *pick something*). This **intent** is both associated to a **frame** (inspired by Frame Semantics [37]) and to the **action** to perform by the target system.

In this approximation, **frames** are the means by which an **intent** –and, thus, an **action**– can be identified from an user request. In a nutshell, **frames** model situations that can be elicited by specific words (`FrameHeads`). These specific words are extracted

[3]e.g. a robot that can *give directions* and that can *pick objects* can perform two skills.

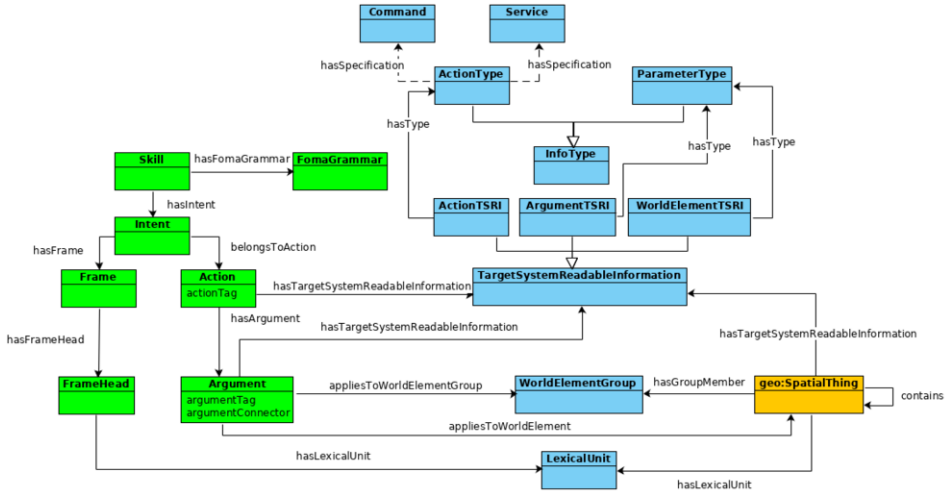

Figure 3. Simple overview of TODODom.

from the user request and, in a dialogue process, the ontology will be in charge of associating them to a **frame** and to their corresponding **intent**. After obtaining the **intent**, it is possible to obtain the **action** to be directed to the target system and its corresponding **arguments** to be fulfilled. The object properties in Frame-Action, besides modelling the relations between classes that have been described above, also allow to define which arguments are compulsory (`hasCoreArgument`) and which are optional (`hasOptionalArgument`) for a specific action. Finally, **actions** and **arguments** have a set of tags associated as data properties, in order to fill the corresponding parametric values in a sentence directed to the user (`outputSentence`).

Frame-Action also includes **Traces** for **skills**, **intents**, **frames**, **actions**, and **arguments**. The **action trace** is the link between TODODom and TODODial, since each user request **trace** is related to the **trace** of the **action** obtained through the user command they represent.

The complete set of classes and object and data properties can be found in the documentation, accessible through the ontology URI in Section 3.4.

TODODW, which covers basic CQ 01, models the set of elements that are available in the scenario. Examples of these elements are spaces, objects or people, depending on the use case. For this, the WGS84 Geo Positioning ontology[4] was considered to reuse. Taking into account the strong dependence of this module on the scenario, this module is practically empty, with an only class, `geo:SpatialThing`, and the relations `dcterms:hasPart` and its inverse `dcterms:hasPartOf`[5], that aim to model the world elements contained by others. This is a module, thus, that must be customized according to the use case. To do so, LOT methodology should be applied, as described in Section 4.

[4] http://www.w3.org/2003/01/geo/wgs84_pos
[5] http://purl.org/dc/terms/

Back to **TODODom**, the role of this upper module is to join its two submodules – described above– through intermediate classes and relations to obtain a complete modelling of the domain of the use case as shown in Figure 3. This TODODom modelling approach allows the dialogue system to associate instructions in natural language to their corresponding commands in the appropriate format for the target system.

These intermediate classes are the **target system readable information** (TSRI), **world element groups** and **lexical units**. The first one models information in a format that is understandable to the target system, which is common to the concepts that have to be provided to the target system. Depending on the **TSRI**, an **information type** (`InfoType`) is associated, which will determine, on the one hand, the implementation of the **action** in regard to the target system (in this case, whether the target system is reachable through a `Service`[6] or simply receives (robot) commands (`Command`)) and, on the other hand, the format of the argument information to be provided to the target system. The second may group similar world elements to relate them to specific arguments, modelling the possible values that an **argument** may have in the context of a specific action. Finally, the third one is common to the concepts that are provided by the user (that is, world elements and actions through **frame heads**). **Lexical units** basically represent the different variants to refer to a **frame head** or a world element.

So as to define the implementation of `Services`, the OWL-S [7] and RESTful Grounding[8] ontologies were reused and reengineered, since they define a semantic markup for web services, including RESTful services.

Ontology Evaluation

According to [38], there are several metrics in the literature to perform ontology evaluation and validation. However, most of these metrics evaluate ontologies from a structural point of view (e.g. whether there are inconsistencies or ill-formed data) or subject-related data (e.g. whether the information in the ontology is correct or whether the domain is fully covered). In the first case, although relevant, these metrics may not be descriptive enough to properly assess the quality of an ontology, whereas in the second case, the metrics must rely on other sources (e.g. gold standards) in order to perform evaluation [39], which in some cases is not viable due to the nature of the ontology to evaluate.

Following the considerations above and the approach and tools described in [31], the following sections present the ontology evaluations from three points of view (**structural metrics**, **design correctedness** and **modularity quality**), that aim to provide a non-biased ontology evaluation. Some discussion is also provided on the possibility of customizing the ontology through module modification, so as to obtain descriptive, comprehensive and objective information regarding the quality of the ontologies[9].

Structural Metrics

This evaluation approach aims to provide some figures describing the data modelled in the ontology, more than assessing its quality [31]. The source of such information is

[6] http://www.daml.org/services/owl-s/1.1/Service.owl

[7] See previous Note.

[8] https://sites.google.com/site/owlsrestful/RESTfulGrounding.owl

[9] The evaluations have been performed in the 2.0 release of TODO: https://github.com/cristinacm/todo/releases/tag/2.0

Table 1. Structural metrics obtained through Protégé's Ontology Metrics tab. Values in parentheses do not consider imported modules.

Ontology	Axioms	Class	OP	DP	Annotation	DL Expressivity
TODODial	560 (29)	66 (0)	22 (3)	13 (0)	17	SHIQ(D)
TODODM	398	52	15	4	17	ALCHI(D)
TODODT	133	14	4	9	17	SHQ(D)
TODODom	442 (376)	35 (26)	40 (20)	16 (13)	17	ALCHIQ(D)
TODODFA	149	8	18	3	17	ALCHIQ(D)
TODODW	17	1	2	0	15	ALI

Table 2. Results of the evaluation on design correctness performed by OOPS!

Ontology	M	I	C	Notes
TODODial	2	1	0	P11, P13, P22
TODODM	2	1	0	P11, P13, P22
TODODT	1	2	0	P04, P11, P13
TODODom	4	1	0	P04, P08, P11, P13, P22
TODODFA	0	1	0	P13
TODODW	2	1	0	P04, P08, P11

Protégé, which includes this specific information in its Ontology Metrics tab. Table 1 provides with the most relevant information in said tab.

Apart from offering statistical measures (e.g., number of axioms, classes and properties), Table 1 also provides information about the expressivity of the ontology at hand. In this sense, due to the nature of the ontologies involved, which aim to model complex relationships between concepts, the presented ontologies show a considerably rich expressivity. However, TODODW is not as rich as the others, which makes sense taking into consideration its out-of-the-box simplicity.

Design Correctness Metrics

So as to assess the design of the ontologies presented in this work, the tool OOPS! [40] is used. This tool checks the ontology to evaluate against a set of 41 pitfalls, classified according to three levels of importance, which are considered to be the most common pitfalls in ontology design.

Table 2 shows the results obtained by OOPS! for each of the modules of the ontology in terms of number of minor (M), important (I) and critical (C) pitfalls. Note that some of the pitfalls detected for TODODial and TODODom are inherited from their submodules.

For the minor pitfalls, the most repeated are **P8**, **P13** and **P22**. **P8** arises when any ***class/property lacks some annotation*** (e.g. description, label). In these ontologies, this pitfall refers to imported classes and properties. On the other hand, **P13** points out the ***lack of inverse object properties***. Due to the purpose of the ontology they are modelled in, the object properties detected do not require an inverse relationship. Finally, **P22** states that ***name conventions are not correctly followed***. Since this pitfall does not indicate the specific elements to be reviewed, all the affected modules have been manually reviewed and all classes and properties follow the same naming conventions.

For the important pitfalls, as it can be seen in the table, the one that all modules share is **P11**, which states that ***properties lack the modelling of domain and range***. In

Table 3. Results of the evaluation on modularity quality

	TODODom		TODODial		TODO		
Metrics	**DFA**	**DW**	**DM**	**DT**	**Dial**	**Dom**	**T2**
Cohesion	**0.01**	**0.0**	**0.07**	**0.03**	**0.04**	**0.004**	*0-0.25*
Encapsulation	**0.96**	**0.62**	**0.98**	**0.95**	**0.99**	**0.98**	*0.75-1*
Coupling	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	*0-0.25*
Redundancy	**0.08**	**0.08**	**0.03**	**0.03**	**0.01**	**0.01**	*0-0.25*
Size	**29**	3	**71**	**27**	**101**	**91**	*10-1103*
Number of axioms	**149**	17	**398**	**133**	**532**	**414**	*46-3954*
Appropriateness	**0.65**	0.01	0.36	**0.55**	-1.0	0.26	*0.51-0.75*
Atomic size	**3.97**	1.67	**4.18**	3.22	**4.0**	**5.18**	*3.42-7.66*
Intramodule distance	**8.0**	**0.0**	**3104.0**	**74.0**	**3178.0**	**62.0**	*0-340833*
Attribute richness	3.75	**0.0**	**1.21**	**1.0**	**1.32**	4.14	*0-3.44*
Inheritance richness	NaN	NaN	**5.88**	**2.2**	**4.46**	**2.0**	*1-6.44*

the scope of this work, this modelling would be problematic and would not be useful for the purpose of the ontologies, so the correction of this pitfall has been discarded by now.

Finally, it is worth noting that no critical pitalls have been observed in any of the modules of TODO. This fact, in combination with the previous considerations for the rest of pitfalls detected, proves that the modules in TODO are correctly designed.

Modularity Quality

Finally, and considering that TODO is a modular ontology, it is of special relevance to evaluate the quality of each of the ontology modules. For this, the approach in [41] is considered to provide a set of comprehensive measures to be able to determine the quality of an ontology module. The work proposes a set of 14 different module types to be used depending on the final usage of the evaluated ontology module, along with some reference metrics and values in order to be able to evaluate if the ontology module is of high quality. In regard to TODO, all modules belong to the T2 type –*Subject domain modules*, which correspond to subdomains inside a large domain [41].

Considering the complexity of the modular hierarchy in TODO, three evaluations have been performed: the quality of TODODFA and TODODW in relation to TODODom, the quality of TODODM and TODODT in relation to TODODial, and the quality of TODODial (including TODODT and TODODM) and TODODom (including TODODFA and TODODW) in relation to TODO, using the TOMM evaluation tool [41]. The results are included in Table 3, which include the relevant metrics to the T2 module types and their reference values. Some of these reference values for the metrics are small cohesion (i.e. "the extent to which entities in a module are related to each other" [41]), coupling (i.e. the degree in which the concepts in a module are related to concepts in other modules) and redundancy (i.e. "the duplication of axioms within a set of ontology modules" [41]) and large encapsulation (i.e. whether a module can be easily replaced by another or modified without side effects), among others.

In Table 3, for TODODom-related metrics, it can be observed that TODODFA achieves the reference values in general, so it can be stated that it is a module of high quality. In the case of TODODW, size-related metrics (size, number of actions, appropriateness and atomic size) do not fit the defined reference values. This is due to the fact that this ontology is intended to be expanded according to the use case and, thus, it has

a reduced size. In the case of TODODial- and TODO- related metrics, the results show that in both cases the implicated modules widely satisfy the established criteria and, thus, are of high quality.

Ontology Customization by Module Modification
One of the advantages of the modularity of TODO is that it is possible to easily modify the ontology by means of specific parts of each module, according to the requirements of the use case. In general, these modifications may be performed in TODODom, since it is the module that is inherently linked to the use case. The clearest example of this customization is TODODW, which, as noted in Section 3.3, must be extended taking into account the scenario the dialogue system will be implemented in, which will define the classes and further relations between them. However, any module from TODO can be easily modified, as shown in the use cases in Section 4.

3.4. Publication Step

Once the ontology has been implemented, it has been made accessible online, both at human- and machine- readable level.

Each of the modules in TODO include all the recommended metadata following Garijo and Poveda-Villalón guidelines [42]. To generate the documentation, WIDOCO (a WIzard for DOCumenting Ontologies) [43], which creates enriched documentation for ontologies, has been used. Furthermore, it is fully compatible with the metadata from the selected guidelines.

The URI for TODO is the following: https://w3id.org/todo. Through this address, the URIs for the rest of the modules can be accessed. This information is also available in TODO's GitHub repository[10].

4. TODO in Use

To assess TODO's usability as core of a task-oriented dialogue system –which is its main motivation–, a validation process has been performed through three industrial use cases for interaction in Spanish: a guide robot, a computerized maintenance management system (CMMS) and a pick-place robot. This validation consists on the definition of TODODW and the instantiation of TODO as a whole, assuring that all the required classes and relations are present. In this process, the suitability of the modularity of the ontology is also reinforced, since each of these solutions have used different configurations of the ontology depending on the necessities of each application[11].

For the modelling of TODODW for each use case, the LOT methodology has been applied: for each use case, two experts in the application field have been interviewed to define the necessities of the use case and recorded as CQs. Based on these CQs, the relevant classes and relations for the Guide and Pick-Place use cases[12] have been defined in the TODODW module[13] (see Table 4 for details).

[10] https://github.com/cristinacm/todo
[11] **Guide robot:** TODO; **CMMS:** TODODom and TODODM; **Pick-place robot:** TODODom.
[12] Due to the characteristics of the CMMS use case, TODODW modelling was not necessary.
[13] Specific TODODW files can be found at Github in the following link: https://git.io/Jnly4.

Table 4. Created classes and relations for TODODW modelling for each use case.

	Guide	Pick-Place
Classes	8	5
Relations	4	-

Table 5. Number of instances for each use case and ontology module.

Use case	Total instances	TODODial instances	TODODom instances	
			Through strategy in [44]	Rest
Guide	589	58	47	484
CMMS	208	58	51	99
Pick-Place	242	N/A	184	58

For the ontology instantiation task, the TODODial population has been performed only once, since it is common for all use cases, and only a specific domain instantiation has been required for each of them. To reduce the manual effort for each of these instantiations, and considering the number of different words that can be used in a natural-language-based interaction to describe the same circumstances, a strategy to semi-automatically obtain this information has been explored [44] and applied to instantiate TODODFA frames, frame heads and lexical units. The rest of the instances are included by mapping the ontology with existing target system data (e.g. personnel databases) and, when necessary, manually. Table 5 shows the main figures.

This validation evidences that the classes and relations in the ontology, identified through expert knowledge, allow to comprehensively model the domain at play and the dialogue-relevant information to successfully enable a natural interaction between workers and industrial systems.

5. Conclusions

This work has presented TODO, a core ontology that is aimed to provide easily-adaptable dialogue systems that allow natural communication between workers and industrial systems in the context of Industry 4.0.

TODO has been developed following a well-defined methodology, which has been specially designed for industrial contexts. First, the requirements of the ontology have been established by describing its motivation and setting them as 10 basic CQs. Considering the requirements, TODO has been implemented using a modular structure, which allows to gather related knowledge in specific modules and to facilitate the extension or modification of said areas of knowledge without affecting the rest of the ontology.

So as to provide information about the quality of TODO and its modules, a set of evaluations have been carried out, which have proven that the modules are expressively rich –which is of special relevance to the area of application–, well designed and of high quality –according to the relevant characteristics of their module type. To reinforce the information provided in the evaluation step, a set of use cases in which TODO has been instantiated and extended where necessary –so as to be used in a dialogue system– have been presented. All in all, the implementation of TODO in these use cases has provided satisfactory results, as common instances for all use cases have been identified and part of the instantiation process can be performed automatically.

Further work includes fine-tuning the dialogue manager to obtain a stable, robust dialogue system using TODO as its core and the definition of a user study to test the whole dialogue system implementation in real industrial settings.

Acknowledgements

This work was partially supported by the Basque Government's Elkartek research and innovation program, projects EKIN (KK-2020/00055) and DeepText (KK-2020/00088).

References

[1] Oborski P. Man-Machine Interactions in Advanced Manufacturing Systems. The International Journal of Advanced Manufacturing Technology. 2004;23(3-4):227–232.
[2] Messner L, Gattringer H, Bremer H. Efficient Online Computation of Smooth Trajectories along Geometric Paths for Robotic Manipulators. In: Multibody System Dynamics, Robotics and Control. Springer; 2013. p. 17–30.
[3] Müller R, Vette M, Scholer M. Inspector Robot–A New Collaborative Testing System Designed for the Automotive Final Assembly Line. Assembly Automation. 2014.
[4] Kildal J, Fernández I, Lluvia I, Lázaro I, Aceta C, Vidal N, et al. Evaluating the UX Obtained from a Service Robot that Provides Ancillary Way-Finding Support in an Industrial Environment. In: Advances in Manufacturing Technology XXXIII: Proceedings of the 17th International Conference on Manufacturing Research, 10-12 Sep 2019, Belfast. vol. 9. IOS Press; 2019. p. 61.
[5] Bugmann G, Pires JN. Robot-by-voice: Experiments on Commanding an Industrial Robot Using the Human Voice. Industrial Robot: An International Journal. 2005.
[6] Veiga G, Pires J, Nilsson K. Experiments with Service-Oriented Architectures for Industrial Robotic Cells Programming. Robotics and Computer-Integrated Manufacturing. 2009;25(4-5):746–755.
[7] Villani V, Pini F, Leali F, Secchi C. Survey on Human–Robot Collaboration in Industrial Settings: Safety, intuitive interfaces and applications. Mechatronics. 2018;55:248 – 266.
[8] Jurafsky D, Martin JH. Speech and Language Processing (Draft). Hentet; 2020.
[9] Ward W, Issar S. Recent Improvements in the CMU Spoken Language Understanding System. Carnegie-Mellon University Pittsburgh, PA School of Computer Science; 1994.
[10] Antonelli D, Bruno G. Human-Robot Collaboration using Industrial Robots. In: 2nd International Conference on Electrical, Automation and Mechanical Engineering. Atlantis Press; 2017. p. 99–102.
[11] Maurtua I, Fernández I, Tellaeche A, Kildal J, Susperregi L, Ibarguren A, et al. Natural Multimodal Communication for Human–Robot Collaboration. International Journal of Advanced Robotic Systems. 2017;14(4):1–12.
[12] Wei Z, Liu Q, Peng B, Tou H, Chen T, Huang XJ, et al. Task-Oriented Dialogue System for Automatic Diagnosis. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers); 2018. p. 201–207.
[13] Goddeau D, Meng H, Polifroni J, Seneff S, Busayapongchai S. A Form-Based Dialogue Manager for Spoken Language Applications. Proceedings of Fourth International Conference on Spoken Language Processing ICSLP '96. 1996;2:701–704 vol.2.
[14] Lee S, Eskenazi M. Recipe For Building Robust Spoken Dialog State Trackers: Dialog State Tracking Challenge System Description. In: Proceedings of the SIGDIAL 2013 Conference. Metz, France: Association for Computational Linguistics; 2013. p. 414–422.
[15] Lee S. Structured Discriminative Model for Dialog State Tracking. In: Proceedings of the SIGDIAL 2013 Conference; 2013. p. 442–451.
[16] Williams JD. Multi-Domain Learning and Generalization in Dialog State Tracking. In: Proceedings of the SIGDIAL 2013 Conference; 2013. p. 433–441.
[17] Mrkšić N, Séaghdha DO, Thomson B, Gašić M, Su PH, Vandyke D, et al. Multi-Domain Dialog State Tracking Using Recurrent Neural Networks. arXiv preprint arXiv:150607190. 2015.
[18] Henderson M, Thomson B, Young S. Deep Neural Network Approach for the Dialog State Tracking Challenge. In: Proceedings of the SIGDIAL 2013 Conference; 2013. p. 467–471.

[19] Chen, Hongshen and Liu, Xiaorui and Yin, Dawei and Tang, Jiliang. A Survey on Dialogue Systems: Recent Advances and New Frontiers. SIGKDD Explor Newsl. 2017 Nov;19(2):25–35.
[20] Aceta C, Kildal J, Fernández I, Soroa A. Towards an Optimal Design of Natural Human Interaction Mechanisms for a Service Robot with Ancillary Way-Finding Capabilities in Industrial Environments. Production & Manufacturing Research. 2021;9(1):1–32.
[21] Suendermann D, Evanini K, Liscombe J, Hunter P, Dayanidhi K, Pieraccini R. From Rule-Based to Statistical Grammars: Continuous Improvement of Large-Scale Spoken Dialog Systems. In: 2009 IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE; 2009. p. 4713–4716.
[22] Altinok D. An Ontology-Based Dialogue Management System for Banking and Finance Dialogue Systems. arXiv preprint arXiv:180404838. 2018.
[23] Wessel M, Acharya G, Carpenter J, Yin M. OntoVPA-an Ontology-Based Dialogue Management System for Virtual Personal Assistants. In: Advanced Social Interaction with Agents. Springer; 2019. p. 219–233.
[24] Searle JR, Kiefer F, Bierwisch M, et al. Speech Act Theory and Pragmatics. vol. 10. Springer; 1980.
[25] Poveda-Villalón M. A Reuse-Based Lightweight Method for Developing Linked Data Ontologies and Vocabularies. In: Extended Semantic Web Conference. Springer; 2012. p. 833–837.
[26] Fernández-López M, Gómez-Pérez A, Juristo N. Methontology: from Ontological Art towards Ontological Engineering. 1997.
[27] Suárez-Figueroa MC. NeOn Methodology for Building Ontology Networks: Specification, Scheduling and Reuse. Informática; 2010.
[28] Hristozova M, Sterling L. An eXtreme Method for Developing Lightweight Ontologies. In: In Workshop on Ontologies in Agent Systems, 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems. Citeseer; 2002. .
[29] Auer S, Herre H. RapidOWL—An Agile Knowledge Engineering methodology. In: International Andrei Ershov Memorial Conference on Perspectives of System Informatics. Springer; 2006. p. 424–430.
[30] Poveda-Villalón M, Fernández-Izquierdo A, García-Castro R. Linked Open Terms (LOT) Methodology. Zenodo; 2019. Available from: https://doi.org/10.5281/zenodo.2539305.
[31] Esnaola-González I, Bermúdez J, Fernández I, Arnaiz A. EEPSA as a Core Ontology for Energy Efficiency and Thermal Comfort in Buildings. Semantic Web. 2021;Pre-press.
[32] Suárez-Figueroa MC, Gómez-Pérez A, Fernández-López M. The NeOn Methodology for Ontology Engineering. In: Ontology Engineering in a Networked World. Springer; 2012. p. 9–34.
[33] Suárez-Figueroa MC, Gómez-Pérez A, Villazón-Terrazas B. How to Write and Use the Ontology Requirements Specification Document. In: OTM Confederated International Conferences "On the Move to Meaningful Internet Systems". Springer; 2009. p. 966–982.
[34] Keet M. An Introduction to Ontology Engineering. vol. 1; 2018.
[35] Rudnicki R, Smith B, Malyuta T, Mandrick W. Best Practices of Ontology Development. CUBRC; 2016.
[36] Bobrow DG, Kaplan RM, Kay M, Norman DA, Thompson H, Winograd T. GUS, a Frame-Driven Dialog System. Artificial Intelligence. 1977;8(2):155–173.
[37] Fillmore CJ, et al. Frame Semantics and the Nature of Language. In: Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech. vol. 280; 1976. p. 20–32.
[38] Villalón MP. Ontology Evaluation: a Pitfall-Based Approach to Ontology Diagnosis; 2016.
[39] Raad J, Cruz C. A Survey on Ontology Evaluation Methods; 2015. .
[40] Poveda-Villalón M, Gómez-Pérez A, Suárez-Figueroa MC. OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation. International Journal on Semantic Web and Information Systems (IJSWIS). 2014;10(2):7–34.
[41] Khan ZC, Keet CM. Dependencies Between Modularity Metrics Towards Improved Modules. In: Blomqvist E, Ciancarini P, Poggi F, Vitali F, editors. Knowledge Engineering and Knowledge Management. Cham: Springer International Publishing; 2016. p. 400–415.
[42] Daniel Garijo and María Poveda-Villalón. A Checklist for Complete Vocabulary Metadata;. https://w3id.org/widoco/bestPractices.
[43] Garijo D. WIDOCO: a Wizard for Documenting Ontologies. In: International Semantic Web Conference. Springer, Cham; 2017. p. 94–102.
[44] Aceta C, Fernández I, Soroa A. Ontology Population Reusing Resources for Dialogue Intent Detection:Generic and Multilingual Approach. Acceptation pending.