Integrating Ontologies and Cognitive Conversational Agents in On2Conv

Zeinab Namakizadeh Esfahani[1], Débora Cristina Engelmann[2], Angelo Ferrando[1], Massimiliano Margarone[3], and Viviana Mascardi[1](✉)

[1] Università degli Studi di Genova, Genova, Italy
n.e.zainab@gmail.com, {angelo.ferrando,viviana.mascardi}@unige.it
[2] Faculdades Integradas de Taquara, Taquara-RS, Brazil
deboraengelmann@faccat.br
[3] SPX LAB, Genova, Italy
m.margarone@spxlab.com

Abstract. Multiagent systems have been successfully used in many domains. Being social, they are expected to communicate with human users in natural language. Nevertheless, the natural interaction between agents and humans is still challenging. Chatbot technologies are a key enabler to boost the communication between humans and software agents, but few technical solutions exist that make the agents' reasoning capabilities easily accessible by a human user via a chatbot and, on the other hand, the chatbot's answers more controllable and explainable. Dial4JaCa is one of such tools. It creates a bridge between Dialogflow and the JaCaMo cognitive-oriented and symbolic AI-based framework: the user's interface is a Dialogflow chatbot allowing the user to interact in natural language, and the backend implementing the reasoning and performing required actions is a JaCaMo agent. However, in Dial4JaCa the consistency between data that feed the JaCaMo agent and those that feed the Dialogflow chatbot must be guaranteed by the developer via an error-prone and tedious manual process. By taking an ontology describing the domain of interest in input and generating both the skeleton for the JaCaMo agent's behaviour and the intents for the Dialogflow chatbot, On2Conv improves Dial4JaCa robustness and reliability, and moves one step towards an explainable integration of agents and chatbots.

Keywords: Conversational agents · Cognitive agents · BDI agents · Ontologies

1 Introduction

With the recent widespread interest in using chatbots in different areas, research on making them as intelligent as possible has gained attention. Powerful chatbot platforms now offer Machine Learning techniques, especially Large Language Models, LLM [30], and in particular Generative Pre-Trained Transformers [24]

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
V. Malvone and A. Murano (Eds.): EUMAS 2023, LNAI 14282, pp. 66–82, 2023.
https://doi.org/10.1007/978-3-031-43264-4_5

such as OpenAI's ChatGPT[1] and GPT-4[2], to detect – besides many other things – the user's intent from an utterance. The GPT hype seems unstoppable: at January 2023, according to a study of the UBS Swiss bank, ChatGPT was estimated to be the fastest growing app ever[3].

The hype also brings issues. On March 14th, 2023, the offices of the European Parliament shared a first draft on General Purpose Artificial Intelligence like ChatGPT, proposing some obligations for the providers of these AI models and responsibilities for the different economic actors involved[4]. The draft suggests that throughout their lifecycle, ChatGPT and similar models will have to undergo external audits testing their performance, predictability, interpretability, corrigibility, safety and cybersecurity in line with the European AI Act's strictest requirements. On March 31st, ChatGPT was banned by the Italian data protection authority over privacy concerns[5]. It was restored on April 28th, but France and Spain had shared similar worries in the meantime.

The concerns of authorities, scientists and citizens about the use of black-box General Purpose Artificial Intelligence and the ongoing debate suggest that the benefits of integrating chatbots – even last-generation ones – with intelligent agents equipped with logic-based reasoning capabilities should be better explored. Indeed, the benefits of this integration might be twofold. On the one hand, users might talk to agents in natural language, removing the interaction barrier that is still preventing the large adoption of intelligent agents – and in particular of cognitive, logic-based ones – by the industry and by people on the street. On the other hand, thanks to the symbolic approach underneath the agent, the user-agent conversation might be controlled, monitored, explained, and steered towards directions implemented in the agent's reasoning rules.

For instance, let us consider a user that asks a health assistant agent *"I am really scared by these three symptoms together: blurred vision, increased thirst and need to urinate often"*; ChatGPT would try to reassure the user by answering *"I understand that these symptoms can be concerning, but it's important not to panic. [...] The most common cause of these symptoms is diabetes, which can be easily diagnosed with a blood test."*[6]. However, the health assistant agent must be aware of its user's gender, age, medications, and it should be able to reason about them and their relations.

Let us suppose that the user is Bob, a male following Fluoxetine therapy. The three symptoms might be side effects of the medication, although they are not frequent ones. The health assistant agent should be aware that Bob takes Fluoxetine because of his illness anxiety disorder and should not mention diabetes as a common cause of these symptoms, at least in its first answer,

1 https://openai.com/blog/chatgpt, accessed on May 2023.
2 https://openai.com/research/gpt-4, accessed on May 2023.
3 https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/, accessed on May 2023.
4 https://www.euractiv.com/section/artificial-intelligence/news/leading-eu-lawmakers-propose-obligations-for-general-purpose-ai/, accessed on May 2023.
5 https://www.bbc.com/news/technology-65139406, accessed on May 2023.
6 Query-Answer experimented on May 5th, 2023.

not to alarm Bob. ChatGPT's *"it's important not to panic"* statement does not seem the best advice for a patient with an anxiety disorder, who might react with panic! Rather, the health assistant agent might report the conversation to the physician in charge of Bob, to keep her updated and let her intervene if it is the case, and keep on interacting with Bob in a reassuring way, without overdramatizing the situation.

If the user is Alice, a 35 years old female following a Clomiphene Citrate therapy, hopefully, the reason for the three symptoms together might be not diabetes but a desired pregnancy. Clomiphene Citrate is a fertility medication that can cause blurred vision, and pregnancy may increase thirst and the need to urinate. The assistant agent might then answer *"Alice, why not trying a pregnancy test today?"*.

These reactions targeted to the user's needs can be possible only if, besides being equipped with natural language understanding capabilities, the agent is also aware of concepts like medications, what they are taken for, their side effects, the user's diseases, their symptoms, and it is able to reason on them to provide a personalised answer.

To move a first step towards tackling the challenges above, we designed, implemented and tested **On2Conv** to translate the domain knowledge represented in an **On**tology into a **Conv**ersational cognitive agent based on Dial4JaCa [10,11]. In Dial4JaCa, the chatbot-like interface towards the user and the agent behind must be fed by manually generated input represented in two different formats. These two representations have to be generated and kept consistent by the human developer, who is hence exposed to an error-prone process.

By using On2Conv, the two representations of the input for the chatbot-like interface and the JaCaMo agent are instead generated in an automatic fashion starting from the same piece of knowledge, the ontology. This approach ensures that "Every piece of knowledge must have a single, unambiguous, authoritative representation within a system" [18] – the ontology in this case – and solves one of the main engineering issues that we experimented while using Dial4JaCa. As a further advantage, being an "an explicit specification of a conceptualization" [15], the ontology where the domain information is stored can be used in the early requirement analysis stages to allow the developers, the domain experts, the clients and the users to reach an agreement about the domain of interest and the relationships among concepts therein.

The structure of the paper is the following: Sect. 2 introduces the background and the works related to On2Conv. Section 3 presents its design and implementation. Section 4 discusses the features of the ontology that feeds On2Conv and shows some experiments carried out in the domestic violence domain. Finally, Sect. 5 concludes by highlighting some future directions.

2 Background and Related Work

Dialogflow [2] is a *lifelike conversational AI with state-of-the-art virtual agents* developed by Google. It allows users to create personal chatbots, namely conversational agents equipped with intents, entities, and fulfillment.

During a conversation between humans, a human speaker can utter different types of sentences, each one with a different intentional meaning. That meaning can be identified as the *intent* of that sentence. In order to explain the map between sentences and intents to the Dialogflow chatbot, the agent's developer should provide examples of sentences that convey that intent, for each intent that is relevant for the application.

The fulfillment is a sort of help from home: if the agent cannot answer messages for some specific intent, those messages are forwarded to an external, specialized source that is waiting. Fulfillment provides a field where the user can insert the URL address of the service to query. The service at that address will be consulted only for those intents that require it; in that case, Dialogflow will wait for the answer and will forward it to the user.
JaCaMo [4] is a framework for Multiagent Programming that combines:

(1) Jason [5], for programming autonomous agents characterised by mentalistic notions like beliefs, goals, desires, intentions, and ability to reason;
(2) CArtAgO [1,27], for programming environment artifacts;
(3) MOISE [16], for programming multiagent organisations.

Jason is a language inspired by the Beliefs-Desires-Intentions (BDI) paradigm [7,26] that implements the logic-based AgentSpeak(L) language [25]. The Jason elements that are more relevant for programming one individual, cognitive agent are:

– Beliefs: the set of facts the agent knows,
– Goals: the set of goals the agent wants to achieve,
– Plans: the set of pre-compiled, operational plans the agent can use to achieve its goals.

Finally, Dial4JaCa [10,11] provides a bridge between Dialogflow or Rasa[7] [3] conversational agents and Jason agents.

As far as the related work is concerned, the strong connection between agents and semantic web technologies, and ontologies in particular, is as old as the semantic web itself [17]. Also, the connection between agents and chatbots is at the basis of the notion of "conversational agents", agents able to conversate with a human user, that started to flourish in the nineties [8,19,23].

When we move to connections between BDI agents and chatbots, however, the literature is scarce and, most often, the BDI paradigm is used as a theoretical framework, rather than as a technological tool.

As an example, the paper by Miliauskas and Dzemydiene [21] presents a non implemented architecture for a BDI chatbot assistant in a travel planning

[7] An open-source framework for building chat and voice-based AI assistants.

domain. Sirocki's Master Thesis [28] aims at aiding 113, the national suicide prevention center for The Netherlands. While the conversational agents design was based upon the BDI architecture, the technological solutions neither adopted any AgentSpeak-like, declarative agent programming language, nor implemented the standard BDI engine. The application domain is however similar to ours, suggesting that chatbots that need to be "psychologically aware and competent" might benefit from being designed and/or implemented as cognitive agents. The paper by Sugumar and Chandra [29] explore factors influencing the adoption of chatbots for financial sectors by emphasising on the role of user desires in addition to human beliefs. The BDI architecture is not used during the chatbot design stages, but only as a framework to represent the factors which the users expect to get from AI technologies for their adoption (*beliefs*), the user's future *desires*, and their *intentions* to adopt chatbots for financial services.

The normative agent system to prevent cyberbullying presented by Bosse and Stam [6] is old, but closer to our approach from a technological point of view. It consists of multiple agents implemented in Jason that control users' norm adherence within virtual societies: the agents continuously monitor the behaviour of the visitors – in particular, their communicative behaviour –, communicate with each other to maintain shared beliefs of the visitors' characteristics, and apply punishments and rewards to influence their behaviour. To the best of our knowledge, however, the most recent works in this research strand are all related to Dial4JaCa and include the implementation of a conversational agent to support hospital bed allocation [9], RV4JaCa (Runtime Verification for JaCaMo) that aims to control the dialogue flow in a MAS [12], and VEsNA (Virtual Environments via Natural language Agents) [14] that enhances the design of virtual environments by exploiting Dialogflow, JaCaMo, and Unity for building the dynamic virtual environment, and letting human users immerse in it.

3 On2Conv Design and Implementation

On2Conv is meant to fill the gap among cognitive agents (in particular, BDI agents implemented in the AgentSpeak(L) language), conversational agents (in particular, chatbots implemented in DialogFlow), and ontologies (in particular, OWL ontologies).

It is implemented in Java using Eclipse Version 2021-12, and Gradle[8] as the builder. It builds on top of JaCaMo, the OWL API[9] as the API to manage ontologies, and Dialogflow ES as the platform to build the chatbot-like interface. On2Conv is available to the research community on GitHub, https://github.com/znesss/Ontology-to-JaCaMo_and_Dialogflow.

Figure 1 shows how to use On2Conv in conjunction with Dial4JaCa, to automatically generate the Dialogflow chatbot and the JaCaMo cognitive agent. At the bottom of the figure, the core element, namely the domain ontology modeled using the OWL language [20], is shown. The ontology feeds both the Dialogflow

[8] https://gradle.org/, accessed on May 2023.
[9] http://owlcs.github.io/owlapi/, accessed on May 2023.

interface, shown at the top of the figure as a cube, via files in JSON format[10], and the JaCaMo back-end, via AgentSpeak (.asl) files. The purple box tagged with "File Generation by On2Conv" represents the On2Conv system that is in charge of creating these files based on the ontology knowledge, ready to be imported into JaCaMo and Dialogflow.

On the JaCaMo side there are two different agents implemented in Jason: the `com_agent` and the `onto_agent`. The first is the agent whose skeleton .asl code is generated by On2Conv, while the second is in charge of interacting with the ontology in order to find out the correct answers based on the recognized intent's parameter values. The `com_agent` plans must be completed by the developer, but their skeleton is coherent by design with the Dialogflow intents, since both are generated starting from the same source of knowledge, namely the ontology.

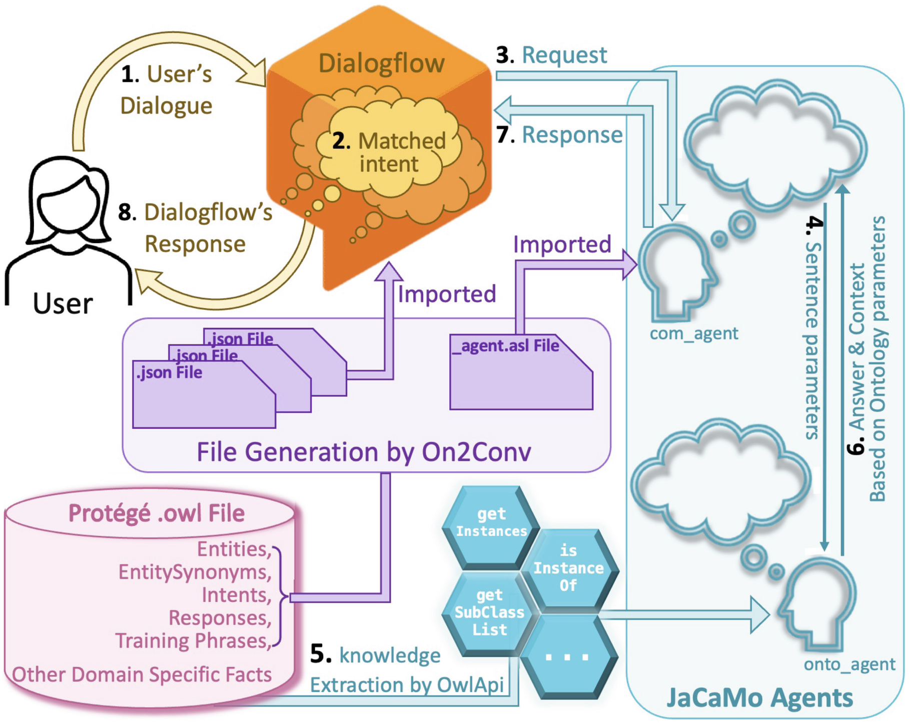


Fig. 1. Methodology to use On2Conv.

After receiving the user's sentence (**1**), Dialogflow automatically matches it with an intent (**2**). Intents are generated by On2Conv based on the ontology knowledge, namely the intent names, training phrases associated with them, and their entities. Hence, we are sure that the matched intent's entities are present

[10] https://www.json.org/json-en.html, accessed on May 2023.

in the ontology for further queries by the JaCaMo agents. The matched intent may have been set to send a request to JaCaMo agents to infer and answer. The existing Dial4JaCa system is represented by the cyan arrows labelled as **3** and **7**, it manages the execution of the response and request services, converting the request to a list of key values. The cyan arrow **5** represents the use of Onto4JaCa[11] [13], which is responsible for providing the methods for the plans that query the ontology, shown by the cyan hexagons. Upon receiving the request, the `com_agent` selects a suitable plan, among those generated by On2Conv and completed by the developer, for sending a message containing necessary parameters to the `onto_agent`, and receives the answer from it. This interaction is shown by numbers **4** and **6**. Dialogflow responds to the user with the received answer, and any possible context provided by the JaCaMo agents (**8**).

On2Conv reads an ontology and produces output to feed the Dialogflow interface, and the JaCaMo agents. Figure 2 shows the interaction between a chatbot developer and On2Conv, where the modules represent On2Conv major methods. On2Conv is equipped with an interface developed by `WindowBuilder` Editor[12], a bi-directional Java GUI designer.

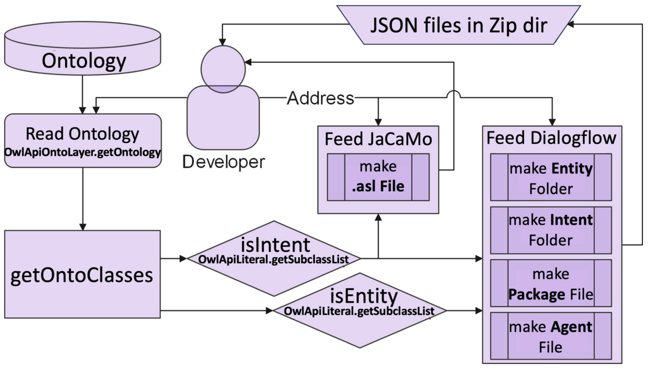


Fig. 2. On2Conv - Chatbot Developer view.

After loading the ontology and choosing a folder to store the JSON and .asl generated from the ontology, the developer will receive messages confirming that the files are created.

The process of generating these files starts with extracting the classes of the ontology as a list and passing them to methods that manage them in order to extract the information needed, and to create the suitable representation in JSON (then compacted into a zip file) and in .asl file.

[11] https://github.com/DeboraEngelmann/Onto4JaCa, accessed on May 2023.
[12] https://www.eclipse.org/windowbuilder/, accessed on May 2023.

As far as the files to feed JaCaMo are concerned, the JaCaMo plans are created with triggering events and contexts corresponding to the intents of ontology and written on the .asl files. The chatbot developer should add the desirable parameter values among those received in the plan's triggering event to the belief base, make the appropriate query, and send it to `onto_agent`. The generated .asl file already contains the plans which build the response message to be sent to Dialogflow, based on the answer received from `onto_agent`.

On the Dialogflow side, the input .zip file consists of JSON format files each carrying distinct information such as entity properties, their entries, training phrases properties, and other items. After importing the zip file, the chatbot developer has to 1. go through all the intents, 2. associate all the entity values of each training phrase with the appropriate entity name in the parameters box, 3. toggle on the required field of parameter[13], 4. set the input context, and 5. turn on the fulfilment option for the intents aimed to redirect to MAS in JaCaMo.

4 Ontology Development and Experiments

Figure 3 shows the interaction between the expert in charge for defining the domain-related concepts that will characterize the ontology ("*the expert*" in the sequel), and the developer of the multiagent system that will take advantage of On2Conv ("*the developer*" in the sequel). The ontology's structure is shown in Fig. 4.

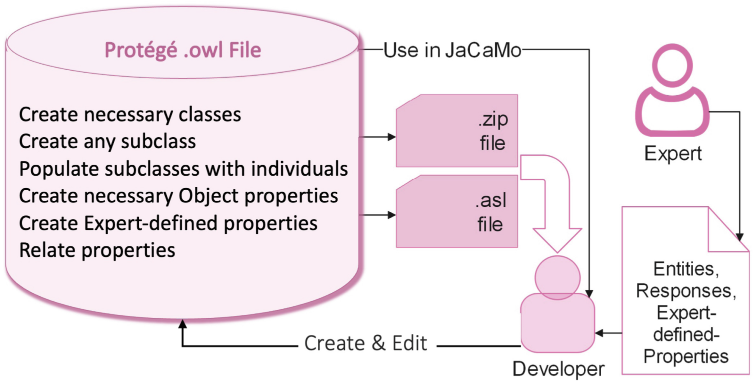
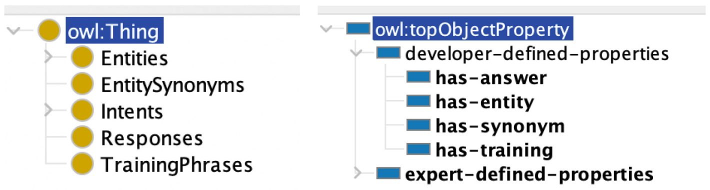

Protégé .owl File
Create necessary classes
Create any subclass
Populate subclasses with individuals
Create necessary Object properties
Create Expert-defined properties
Relate properties
Use in JaCaMo
.zip file
.asl file
Expert
Entities, Responses, Expert-defined-Properties
Create & Edit
Developer

Fig. 3. Ontology Development.

The creation of the ontology is likely to be an iterative process, requiring interactions between the *expert* and the *developer*, and may follow some well known ontology engineering methodology like the Ontology Development 101

[13] This step should be done only in case there are plans in `onto_agent` on JaCaMo side attempting to unify the literal values with variables.



(a) Classes. (b) Object Properties.

Fig. 4. Ontology Structure.

guide [22]. The second step proposed in that guide, is "consider reusing existing ontologies". Ontology sharing and reuse is one of the main motivations for On2Conv, besides improving the Dial4JaCa engineering. Given that an ontology to be used as input for On2Conv must respect some constraints related to its structure and hierarchy, we provide further and ad-hoc instructions on On2Conv ontology development (or adaptation for reuse) at https://github.com/znesss/Ontology-to-JaCaMo_and_Dialogflow/wiki#onto-development.

There are two sources for the information to be integrated into the ontology. The first is a dialogue chart that must be designed by the *developer*, with the help of the *expert*, considering as many scenarios as possible. This chart may contain simple short conversations as "User: Hello, how do I look today?", "Bot: You look fantastic, Dalia." or rather complex ones as "User: Hello, what do you suggest me to read?", "Bot: Tell me more about your list of favourite genres and writers, Dalia.", "User: The last book I enjoyed reading was Quaderno proibito".

This dialogue chart helps the *developer* to identify topics (`Intents`) the user's sentence has to be matched with, in order for the agent to provide a suitable answer. Such sentences are categorised as the training phrases of that intent. During this process, entities detected to be used as a word to help the answer-generation are marked to be later added to their own proper class inside the Entity class.

The second source of information is the knowledge coming from the *expert*, that will feed the `Entities` class. For instance, if the *expert* provides the fact that "Setting clear goals, making plans, eliminating distractions, and taking breaks can help people focus and stay motivated.", the *developer* might create an `Entity` named `MotivationBooster` and include `taking_breaks` as instance.

After making the list of Intents, Entities, Training phrases, Responses, and Entity Synonyms, the developer has to create the classes with the same names and include their items as the subcategories or individuals. Following that, the properties linking these classes have to be defined. An object property must relate individuals to individuals only. Therefore, in order to relate each intent to multiple training phrases, the *developer* has to create individuals inside each intent class. These individuals may represent subcategories of their parent class, and in addition to allowing the *developer* to incorporate more *expert* knowledge by defining a more comprehensive ontology, their use is enabling the developer to relate them with training phrases through the property has-training. Table 1 shows the domain and range of the four properties that the *developer* must instantiate.

Table 1. Developer-defined-properties.

Name	Domain	Range
`has-training`	Intents	TrainingPhrases
`has-entity`	TrainingPhrases	Entities
`has-synonym`	Entities	EntitySynonyms
`has-answer`	Intents	Responses

As an example of ontology to be used to feed On2Conv we introduce `donna-MAMi` for creating a motivational agent for women experiencing domestic violence. The `donnaMAMi` structure is shown in form of tree in Fig. 5. Not all the classes are shown for lack of space.

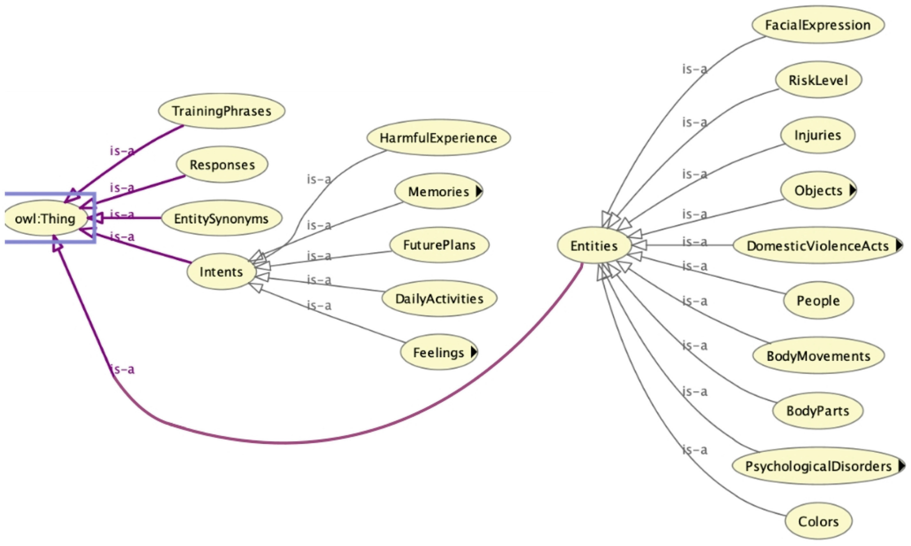
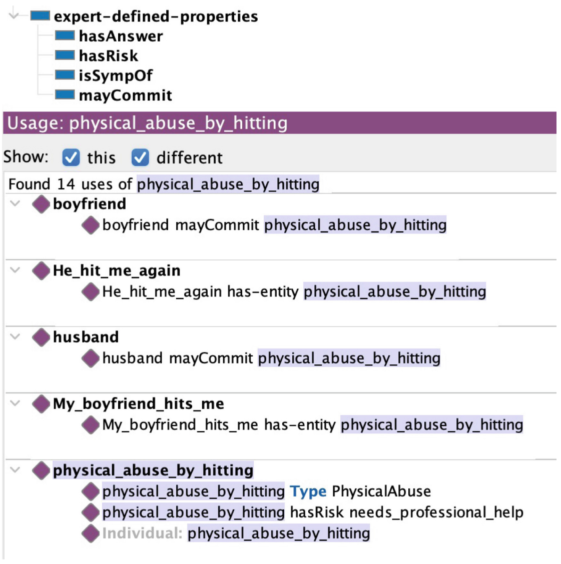


Fig. 5. Tree-like view of `donnaMAMi`.

In `donnaMAMi`, Intents include `DailyActivities`, `Feelings`, `FuturePlans`, `Memories`, hence topics that may be dealt with during a conversation between a woman and the motivational agent; some Intents are broken into sub-classes and may have instances (individuals). Also, some Intents are more related with the domestic violence domain, such as `HarmfulExperience`.



```
expert-defined-properties
    hasAnswer
    hasRisk
    isSympOf
    mayCommit
Usage: physical_abuse_by_hitting
Show: ☑ this ☑ different
Found 14 uses of physical_abuse_by_hitting
boyfriend
    boyfriend mayCommit physical_abuse_by_hitting
He_hit_me_again
    He_hit_me_again has-entity physical_abuse_by_hitting
husband
    husband mayCommit physical_abuse_by_hitting
My_boyfriend_hits_me
    My_boyfriend_hits_me has-entity physical_abuse_by_hitting
physical_abuse_by_hitting
    physical_abuse_by_hitting Type PhysicalAbuse
    physical_abuse_by_hitting hasRisk needs_professional_help
    Individual: physical_abuse_by_hitting
```

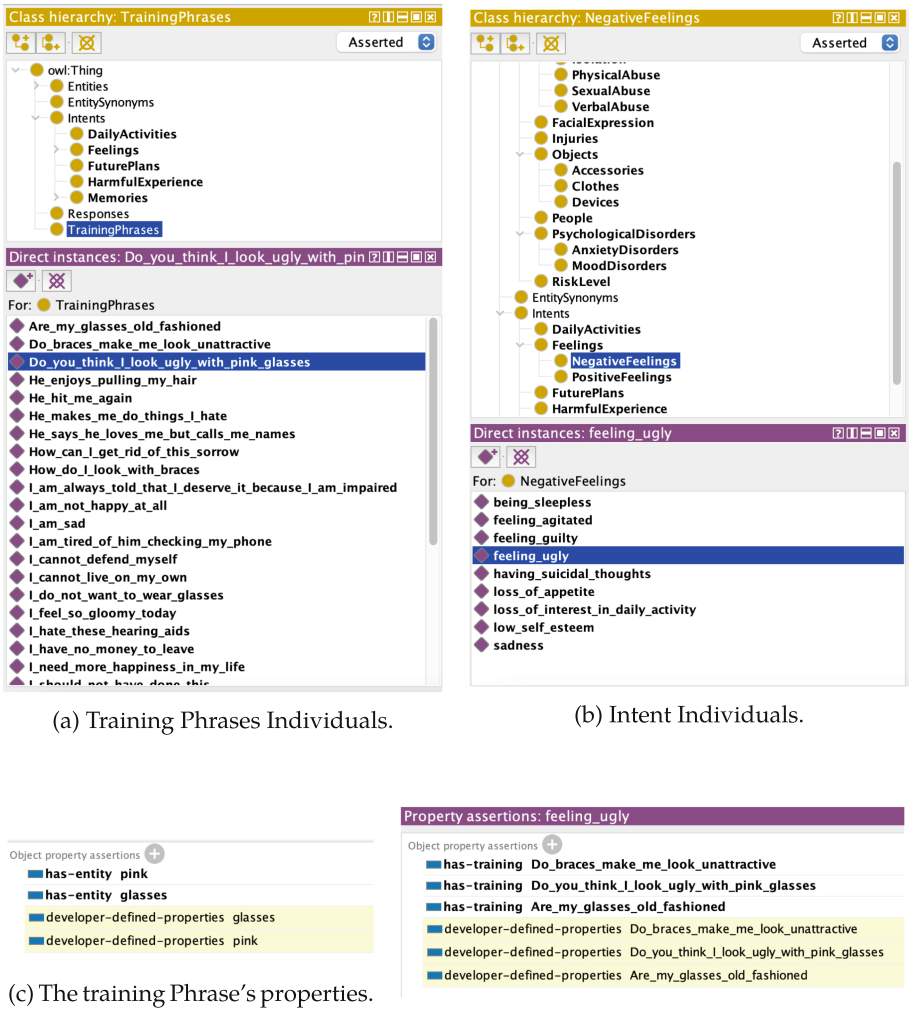
Fig. 6. `expert-defined-properties`.

Entities are designed to incorporate the domain dependent concepts. Take the entity `Emotional Abuse` for example. Its instances come from the expert that should be a psychologist with experience in the domestic violence domain, and are used to conclude the degree of the abuse, in order to generate acceptable response and/or take appropriate action. `DomesticViolenceActs`, `PsychologicalDisorders`, and `RiskLevel` are the entities developed entirely based on the psychology domain texts that we consulted in order to design the `donnaMaMi` ontology, given the absence of a domain *expert* in the team.

Figure 6 shows an example of how the *expert* might define relations. For example, both a husband and a boyfriend may commit physical abuse by hitting, and some examples of sentences that are related to this kind of abuse are "He hit me again", represented in the ontology as an individual identified by the name `He_hit_me_again`, and "My boyfriend hits me".

Further examples of relations between the individuals for the daily conversations are provided in Fig. 7. The `expert-defined-properties` shown in Fig. 6 have `hasAnswer`, `hasRisk`, `isSympOf`, `mayCommit` as sub-properties, and the list might be further expanded. For example, `hasRisk` links an individual of `DomesticViolenceActs` (a subclass of `Entities` class) to an individual of `RiskLevel` which is an `Entities`' sub-class as well. This information is used by the reasoning system to find out the risk level of specific a domestic violence act.

It is also possible to include in the `expert-defined-properties`, the properties linking an `Intents` individual to an `Entities` individual, or vice-versa. For instance in `donnaMAMi` ontology, `isSympOf`'s domain is `NegativeFeelings`, and its range is `PsychologicalDisorder` which is referring to the fact that any



Are_my_glasses_old_fashioned
Do_braces_make_me_look_unattractive
Do_you_think_I_look_ugly_with_pink_glasses
He_enjoys_pulling_my_hair
He_hit_me_again
He_makes_me_do_things_I_hate
He_says_he_loves_me_but_calls_me_names
How_can_I_get_rid_of_this_sorrow
How_do_I_look_with_braces
I_am_always_told_that_I_deserve_it_because_I_am_impaired
I_am_not_happy_at_all
I_am_sad
I_am_tired_of_him_checking_my_phone
I_cannot_defend_myself
I_cannot_live_on_my_own
I_do_not_want_to_wear_glasses
I_feel_so_gloomy_today
I_hate_these_hearing_aids
I_have_no_money_to_leave
I_need_more_happiness_in_my_life
I_should_not_have_done_this

(a) Training Phrases Individuals.

being_sleepless
feeling_agitated
feeling_guilty
feeling_ugly
having_suicidal_thoughts
loss_of_appetite
loss_of_interest_in_daily_activity
low_self_esteem
sadness

(b) Intent Individuals.

has-entity pink
has-entity glasses
developer-defined-properties glasses
developer-defined-properties pink

(c) The training Phrase's properties.

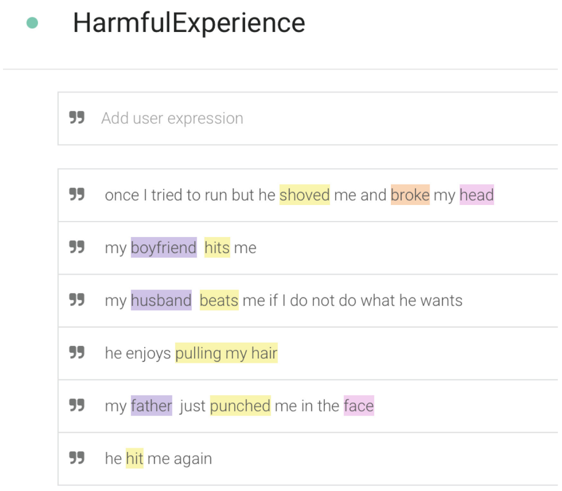
has-training Do_braces_make_me_look_unattractive
has-training Do_you_think_I_look_ugly_with_pink_glasses
has-training Are_my_glasses_old_fashioned
developer-defined-properties Do_braces_make_me_look_unattractive
developer-defined-properties Do_you_think_I_look_ugly_with_pink_glasses
developer-defined-properties Are_my_glasses_old_fashioned

(d) the Intent's properties.

Fig. 7. `donnaMAMi` Ontology Example.

negative feeling the user is talking about may lead to a psychological disorder. Therefore, the part of knowledge provided by the *expert* can be related to the topics of the conversations going on between the chatbot and the user devised by the *developer*.

The sentences inserted in the `donnaMAMi` instances are used to automatically generate training sentences for Dialogflow as shown in Fig. 8.


HarmfulExperience

Add user expression

once I tried to run but he shoved me and broke my head

my boyfriend hits me

my husband beats me if I do not do what he wants

he enjoys pulling my hair

my father just punched me in the face

he hit me again

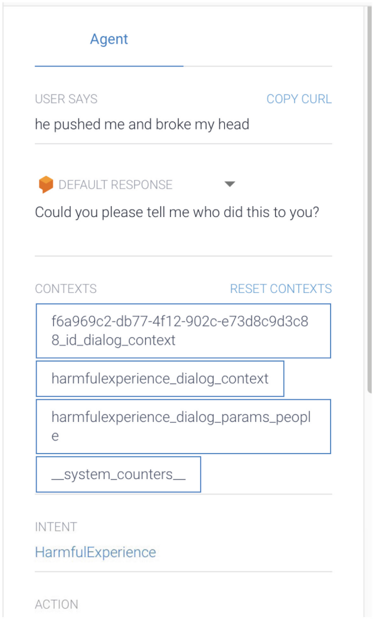
Fig. 8. Training phrases for the Dialogflow HarmfulExperience Intent generated starting from the `donnaMAMi` ontology.

On the other hand, Listing 1.1 shows the Jason plan of the `com_agent` agent, named `donnaMAMi_com_agent` in this instance of the MAS, also automatically generated by On2Conv and then edited by the developer to add domain dependent behaviour. As anticipated in Sect. 3, the `com_agent` acts as an interface between Dialogflow and the ontology. The plan shown in Listing 1.1 deals with the case of having recognized "HarmfulExperience" as the intent of the user's sentence (second line, representing the context of the Jason plan). The `com_agent` actions in the plan's body are aimed at sending a request for the correct answer to the `onto_agent`, named the `donnaMAMi_onto_agent` in this MAS (last line), after the parameters of the intent are properly managed. The `donnaMAMi_onto_agent` is in charge of looking for the correct answer in the ontology. At this stage of the On2Conv development, we were mainly focused on the On2Conv implementation and on its coherence. Although the plan shown in Listing 1.1 mainly implements a simple reactive behaviour, without any logical reasoning, Jason natively supports logic-based reasoning on the agents' beliefs that are represented in an explicit, symbolic way. The examples of sophisticated integration of knowledge and deduction of new facts presented in the Introduction can indeed be implemented in Jason thanks to its support to Prolog-like rules. While we do not claim that this implementation would be effortless, we believe that it would bring significant advantages in terms of code readability, shareability, and explainability, supporting a transparent approach to modern Artificial Intelligence.

```
+!responder(RequestedBy, RespId, IntentName, Params, Contexts)
    : (IntentName == "HarmfulExperience")
<-  !delprevparams;
     for ( .member(X,Params) ) { +X; };
      ?param("PhysicalAbuse",ABUS); ?param("BodyParts",BD);
      ?param("Injuries",INJ); ?param("People",PPL);
     .send(donnaMAMi_onto_agent,askOne,
         answerHarmfulExperience(ABUS, BD, INJ, PPL, RespId)).
```

Listing 1.1. Jason plan for the donnaMAMi_com_agent edited after being generated starting from the `donnaMAMi` ontology.

Finally, in Fig. 9, two different scenarios for a specific context are shown. The context is `needs_professional_help` when "any" person "has done/does/is doing" any "physical harm" to the user. In the first scenario (Fig. 9a), if the user does not provide the necessary information (in the tested case: person), she will be prompted by the chatbot to mention it, to take the best action based on the answer. The second scenario (Fig. 9b) copes with the situation where an injury is detected in the uttered sentence of the user. In this case, the priority is to tell the user to call the ambulance. Besides from taking care of the context generation, the type of injury is extracted, therefore the agent answers with "Have you called the ambulance?" to suggest the user to immediately do it, in a gentle, non assertive way.



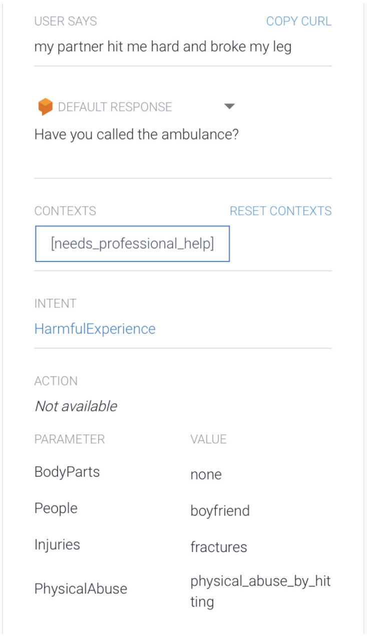
he pushed me and broke my head

Could you please tell me who did this to you?

f6a969c2-db77-4f12-902c-e73d8c9d3c88_id_dialog_context
harmfulexperience_dialog_context
harmfulexperience_dialog_params_people
__system_counters__

HarmfulExperience

(a) First scenario.



my partner hit me hard and broke my leg

Have you called the ambulance?

[needs_professional_help]

HarmfulExperience

Not available

PARAMETER	VALUE
BodyParts	none
People	boyfriend
Injuries	fractures
PhysicalAbuse	physical_abuse_by_hitting

(b) Second scenario.

Fig. 9. Experiments: generating the correct context for the conversation based on what the user says.

5 Conclusions and Future Work

In this paper, the On2Conv tool to generate the files for feeding the Dialogflow and JaCaMo agents starting from information stored in an ontology has been presented. On2Conv adds robustness to the agents communicating through Dial-4JaCa, and makes the development process faster and less error-prone eliminating duplication of information, as suggested by the "Don't repeat yourself" (DRY) software engineering principle.

The feasibility of the methodology using On2Conv is explored through the development of the `donnaMAMi` ontology and its exploitation to create a MAS. The `donnaMAMi` MAS has been tested by the authors with sample sentences that a user may utter in a simplified world. Although no tests were carried out in a real-world environment with any possible sentence about domestic violence, the test sentences were chosen wisely to cover as many aspects as possible in order for the MAS reasoning power to be evaluated.

To improve the donnaMAMi ontology and to move the experiment outside the boundaries of academia, we are currently interacting with the SAVE THE WOMAN Italian association of social promotion[14].

Born in 2020, SAVE THE WOMAN is responsible for promoting and disseminating digital solutions against gender-based violence. One of these solutions is the **NONPOSSO**PARLARE (**ICANNOT**TALK) chatbot, developed by SPX under the guidance of one of the authors of this paper. In the last two years **NONPOSSO**PARLARE has been integrated in more than twenty women assistance portals, collecting – in a fully anonymous, GDPR-compliant way – valuable information on how victims use digital tools to ask help. We are currently evaluating how we can integrate the **NONPOSSO**PARLARE chatbot with ontological knowledge and with reasoning capabilities, thanks to On2Conv.

During the development of On2Conv and of its experimentation we relied on our own sensitivity and common sense to address the psychological aspects related with domestic violence. With the help of the SAVE THE WOMAN on the field staff, however, the `donnaMAMi` ontology and the agents' reasoning mechanism might be made more realistic and once injected into **NONPOSSO**PARLARE, they might make it more competent and solid.

References

1. Cartago. https://cartago.sourceforge.net/
2. Dialogflow documentation. https://cloud.google.com/dialogflow/docs/
3. Introduction to rasa open source & rasa pro. https://rasa.com/docs/rasa/
4. Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A., Santi, A.: Multi-agent oriented programming with JaCaMo. Sci. Comput. Program. **78**(6), 747–761 (2013). https://doi.org/10.1016/j.scico.2011.10.004
5. Bordini, R.H., Hübner, J.F., Wooldridge, M.J.: Programming Multi-agent Systems in AgentSpeak using Jason. Wiley, Hoboken (2007)

[14] https://www.savethewoman.org/, accessed on May 2023.

6. Bosse, T., Stam, S.: A normative agent system to prevent cyberbullying. In: Boissier, O., Bradshaw, J., Cao, L., Fischer, K., Hacid, M. (eds.) Proceedings of the 2011 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2011, Campus Scientifique de la Doua, Lyon, France, 22–27 August 2011, pp. 425–430. IEEE Computer Society (2011). https://doi.org/10.1109/WI-IAT.2011.24
7. Bratman, M.: Intention, Plans, and Practical Reason. Harvard University Press, Cambridge, MA (1987)
8. Cassell, J., et al.: Embodiment in conversational interfaces: Rea. In: Williams, M.G., Altom, M.W. (eds.) Proceeding of the CHI 1999 Conference on Human Factors in Computing Systems: The CHI is the Limit, Pittsburgh, PA, USA, 15–20 May 1999, pp. 520–527. ACM (1999). https://doi.org/10.1145/302979.303150
9. Engelmann, D.C., Cezar, L.D., Panisson, A.R., Bordini, R.H.: A conversational agent to support hospital bed allocation. In: Britto, A., Delgado, K.V. (eds.) Intelligent Systems - 10th Brazilian Conference, BRACIS 2021, Virtual Event, 29 November – 3 December 2021, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13073, pp. 3–17. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-91702-9_1
10. Engelmann, D.C., et al.: Dial4jaca - a demonstration. In: Dignum, F., Corchado, J.M., de la Prieta, F. (eds.) Advances in Practical Applications of Agents, Multi-Agent Systems, and Social Good. The PAAMS Collection - 19th International Conference, PAAMS 2021, Salamanca, Spain, 6–8 October 2021, Proceedings. Lecture Notes in Computer Science, vol. 12946, pp. 346–350. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-85739-4_29
11. Engelmann, D.C., et al.: Dial4jaca - a communication interface between multi-agent systems and chatbots. In: Dignum, F., Corchado, J.M., de la Prieta, F. (eds.) Advances in Practical Applications of Agents, Multi-Agent Systems, and Social Good. The PAAMS Collection - 19th International Conference, PAAMS 2021, Salamanca, Spain, 6–8 October 2021, Proceedings. Lecture Notes in Computer Science, vol. 12946, pp. 77–88. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-85739-4_7
12. Engelmann, D.C., Ferrando, A., Panisson, A.R., Ancona, D., Bordini, R.H., Mascardi, V.: Rv4jaca - towards runtime verification of multi-agent systems and robotic applications. Robotics **12**(2), 49 (2023). https://doi.org/10.3390/robotics12020049
13. Engelmann, D.C.: Intentional dialogues in multi-agent systems based on ontologies and argumentation. Ph.D. thesis, Pontifícia Universidade Católica do Rio Grande do Sul and University of Genoa (Double-degree) (2023)
14. Gatti, A., Mascardi, V.: Vesna, a framework for virtual environments via natural language agents and its application to factory automation. Robotics **12**(2), 46 (2023). https://doi.org/10.3390/robotics12020046
15. Gruber, T.R.: A translation approach to portable ontology specifications. Knowl. Acquis. **5**(2), 199–220 (1993)
16. Hannoun, M., Boissier, O., Sichman, J.S., Sayettat, C.: MOISE: an organizational model for multi-agent systems. In: Monard, M.C., Sichman, J.S. (eds.) IBERAMIA/SBIA -2000. LNCS (LNAI), vol. 1952, pp. 156–165. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44399-1_17
17. Hendler, J.A.: Agents and the semantic web. IEEE Intell. Syst. **16**(2), 30–37 (2001). https://doi.org/10.1109/5254.920597
18. Hunt, A., Thomas, D.: The Pragmatic Programmer: From Journeyman to Master. Addison-Wesley, Boston [etc.] (2000). http://www.amazon.com/The-Pragmatic-Programmer-Journeyman-Master/dp/020161622X

19. Massaro, D.W., Cohen, M.M., Daniel, S., Cole, R.A.: Chapter 7 - developing and evaluating conversational agents. In: Hancock, P. (ed.) Human Performance and Ergonomics, pp. 173–194. Handbook of Perception and Cognition (Second Edition), Academic Press, San Diego (1999). https://doi.org/10.1016/B978-012322735-5/50008-7, https://www.sciencedirect.com/science/article/pii/B9780123227355500087
20. McGuinness, D.L., Van Harmelen, F., et al.: OWL web ontology language overview. W3C Recommendation **10**(10), 2004 (2004)
21. Miliauskas, A., Dzemydiene, D.: An approach to designing belief-desire-intention based virtual agents for travel assistance. In: Lupeikiene, A., Matulevicius, R., Vasilecas, O. (eds.) Joint Proceedings of Baltic DB&IS 2018 Conference Forum and Doctoral Consortium co-located with the 13th International Baltic Conference on Databases and Information Systems (Baltic DB&IS 2018), Trakai, Lithuania, 1–4 July 2018. CEUR Workshop Proceedings, vol. 2158, pp. 94–103. CEUR-WS.org (2018). https://ceur-ws.org/Vol-2158/paper10.pdf
22. Noy, N.F., McGuinness, D.L.: Ontology development 101: A guide to creating your first ontology (2001). https://protege.stanford.edu/publications/ontology_development/ontology101.pdf
23. Nugues, P., Godéreaux, C., El Guedj, P.O., Revolta, F.: A conversational agent to navigate in virtual worlds. In: Proceedings Dialogue Management in Natural Language Systems. Twente Workshop on Language Technology, vol. 11, pp. 23–33 (1996)
24. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training (2018). https://gwern.net/doc/www/s3-us-west-2.amazonaws.com/d73fdc5ffa8627bce44dcda2fc012da638ffb158.pdf
25. Rao, A.S.: Agentspeak(l): BDI agents speak out in a logical computable language. In: de Velde, W.V., Perram, J.W. (eds.) Agents Breaking Away, 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, Eindhoven, The Netherlands, 22–25 January 1996, Proceedings. Lecture Notes in Computer Science, vol. 1038, pp. 42–55. Springer, Cham (1996). https://doi.org/10.1007/BFb0031845
26. Rao, A.S., Georgeff, M.P., et al.: BDI agents: from theory to practice. In: Icmas, vol. 95, pp. 312–319 (1995)
27. Ricci, A., Piunti, M., Viroli, M., Omicini, A.: Environment programming in CArtAgO. In: El Fallah Seghrouchni, A., Dix, J., Dastani, M., Bordini, R.H. (eds.) Multi-Agent Programming, pp. 259–288. Springer, Boston, MA (2009). https://doi.org/10.1007/978-0-387-89299-3_8
28. Sirocki, J.: Design and evaluation of a conversational agent model based on stance and BDI providing situated learning for triage-psychologists in the helpline of 113 suicide prevention (2019). master Thesis, Delft University of Technology
29. Sugumar, M., Chandra, S.: Do i desire chatbots to be like humans? Exploring factors for adoption of chatbots for financial services. J. Int. Technol. Inf. Manag. **30**(3), 38–77 (2021)
30. Vaswani, A., et al.: Attention is all you need. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4–9 December 2017. Long Beach, CA, USA, pp. 5998–6008 (2017). https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html