

Position paper

An ontology of online user feedback in software engineering

Itzel Morales-Ramirez ^{a,b,*}, Anna Perini ^a and Renata S.S. Guizzardi ^c

^a *Software Engineering Research Unit, Fondazione Bruno Kessler, IRST, Trento, Italy*
E-mails: imramirez@fbk.eu, perini@fbk.eu

^b *University of Trento, Trento, Italy*

^c *Ontology and Conceptual Modeling Research Group, UFES, Espírito Santo, Brazil*
E-mail: rguizzardi@inf.ufes.br

Abstract. Online user feedback is principally used as an information source for evaluating customers' satisfaction for a given goods, service or software application. The increasing attitude of people towards sharing comments through the social media is making online user feedback a resource containing different types of valuable information. The huge amount of available user feedback has drawn the attention of researchers from different fields. For instance, data mining techniques have been developed to enable information extraction for different purposes, or the use of social techniques for involving users in the innovation of services and processes. Specifically, current research and technological efforts are put into the definition of platforms to gather and/or analyze multi-modal feedback. But we believe that the understanding of the type of concepts instantiated as information contained in user feedback would be beneficial to define new methods for its better exploitation. In our research, we focus on online explicit user feedback that can be considered as a powerful means for user-driven evolution of software services and applications. Up to our knowledge, a conceptualization of user feedback is still missing. With the purpose of contributing to fill up this gap we propose an ontology, for explicit online user feedback that is founded on a foundational ontology and has been proposed to describe artifacts and processes in software engineering. Our contribution in this paper concerns a novel user feedback ontology founded on a Unified Foundational Ontology (UFO) that supports the description of analysis processes of user feedback in software engineering. We describe the ontology together with an evaluation of its quality, and discuss some application scenarios.

Keywords: User feedback ontology, communication ontology, speech act theory, user feedback analysis, requirements engineering, software engineering

Accepted by: Nicola Guarino

1. Introduction

In many different domains stakeholders' feedback is key for improving offered services and products, or more generally to enact their evolution. In the teaching domain, feedback is extensively used and plays a crucial role in the advisor–student relation, when a student receives comments about her performance on assigned scholar activities (i.e., writing a technical report) in order to improve or correct her current academic performance (Debusse & Lawley, 2015). In marketing, the management of user feedback is

*Corresponding author: Itzel Morales-Ramirez, Fondazione Bruno Kessler, Via Sommarive 18, I-38123 Trento-Povo, Italy.
E-mail: imramirez@fbk.eu.

mainly applied to evaluate customers' satisfaction for a given goods, service or software application (Proynova & Paech, 2013; Moghaddam, 2015).

The increasing attitude of people towards sharing comments through the social media or through mobile applications, indeed, is making online user feedback a resource of valuable information for all types of stakeholders. These stakeholders are, for instance, companies, service providers or software developers but also other end users that may like to get an advice on the quality of a service or product from previous experiences. While customer feedback tend to report customer's evaluation in terms of quality-price-experience in comparison to other products, recommendation to people to buy the product (or not) and/or a summary of the product's features, online user feedback regarding a software is given by users who usually report technical details about the software, experience of use, and the most important characteristic refers to the report of bugs or the request of new features.

Concretely, user feedback can be considered as a reaction of users, which roots in their perceived quality of experience. This reaction can be implicitly expressed in the behavior of the user who is constantly accessing a service or gives up in using it, or be explicitly expressed by rating the software application performance and functionality, by sending a report about an error which occurred, or by suggesting new features.

Another important difference between implicit and explicit feedback is that the latter moves from the user intention to communicate to someone something based on their beliefs and goals and sometimes move the audience to act consequently.

In our research, we focus on online explicit user feedback (that we refer simply as "user feedback" from now on) in the context of software services and applications, which can represent a key information source for requirements driven evolution of software services and applications. Indeed, the huge amount of user feedback available through social media or other online communication channels, e.g. App stores, forums, mailing lists, has drawn the attention of researchers from different fields, e.g., machine learning, big data, social networks, and with different purposes, e.g., enrich innovation process, evaluate customer experience, develop recommender systems (Bosch-Sijtsema & Bosch, 2014; Ordenes et al., 2014; Masli & Terveen, 2014; Jawaheer et al., 2014).

Current research and technological efforts are put into the definition of platforms to gather multi-modal feedback (Schneider, 2011; Seyff et al., 2010, 2014). For example, Seyff et al. (2014) provide voice recording and image capturing as a means for collecting feedback, in addition to textual messages. Approaches for a continuous communication of feedback to software engineers are investigated by Maalej et al. (2009). Another research direction concerns the analysis of user feedback. For instance, the exploitation of natural language processing (NLP) techniques to support the analysis of textual feedback is considered in recent research, such as (Carreño & Winbladh, 2013; Ordenes et al., 2014; Guzman & Maalej, 2014; Jawaheer et al., 2014; Reforgiato Recupero et al., 2015).

In our opinion, a definition of the concepts that pertain to the context of user feedback, which would enable a shared understanding of this context, is still missing. For instance, a clear understanding of what type of information is contained in user feedback, and respectively what type of information can be extracted from it, would be beneficial to define and compare methods and tools to be used for the analysis of feedback.

These considerations motivate us to formulate the following research question: *What are the concepts that may support a framework for automating the analysis of user feedback?*

To answer it, we propose an ontology of user feedback that is founded on already existing ontologies, which describe artifacts and processes in software engineering. The advantages of having an ontology are several, and in our case the most important are: (a) a better understanding of the problem context, i.e.,

user feedback; (b) ontologies represent a scrutiny of all the concepts (including synonyms) that must be taken into account, e.g., user comment is the same as user feedback; and (c) ontologies enable an inter-operation of different techniques, for example among systems that use different terminology, an ontology can provide a translation between the concepts used between the diverse methods, paradigms, languages or software tools implemented.

The considered background knowledge to develop the user feedback ontology is the Speech Act Theory (SAT), which has been proposed in the research field of philosophy of language, and the communication ontology developed as an extension of the Unified Foundational Ontology (UFO) (Oliveira et al., 2007). Specifically, Austin and Searle with SAT, claim that when a person says something she/he attempts to communicate certain things to the addressee, which affect either their beliefs and/or their behavior (Austin, 1962; Searle, 1969).

In a previous work, we described the methodology we followed for the ontology development (Guizzardi et al., 2014), which takes advantage of a goal oriented approach described by Fernandes et al. (2011) for defining the competency questions to be answered by the ontology and the concepts to be considered for its scope. Moreover, in another work we introduced the ontology at an initial stage and how we built it (Morales-Ramirez et al., 2014b), by exploiting the Unified Foundational Ontology (UFO) by Guizzardi (2005).

The novel contributions presented in this paper are three-fold:

- Present the complete user feedback ontology modeled with OntoUML, which is a modeling language whose metamodel is an ontologically well-founded UML extension based on UFO;
- Illustrate the application of the ontology in different scenarios concerning a manual semantic analysis of online user feedback in software application domains, expressed through textual message such as the following example: “I got some problems with the configuration... hopefully someone... could help me out”; and
- Describe how we evaluate the proposed ontology by following the guidelines developed by Sabou and Fernandez (2012).

The remainder of this paper has the following structure. In Section 2 the background of the work is presented. The proposed user feedback ontology is described in Section 3. We illustrate its applicability in Section 4 and present the evaluation of the ontology in terms of quality of modeling in Section 5. The related work is presented in Section 6 while in Section 7 we present concluding remarks and introduce further application purpose of the ontology along with future work directions.

2. Background

The background knowledge of our research includes theories and a foundational ontology, as illustrated in the overview in Fig. 1. The upper region refers to the *problem space* of our research, which includes concepts defined in the literature about Feedback Cycle and Requirements Engineering for the purpose of Software Evolution. From this literature we revise some definitions regarding user feedback and borrow concepts for our ontology. The bottom region of the figure refers to the *solution space* and includes a well-founded communication ontology. A foundational ontology describes the nature and essence of basic concepts present in the real world that are familiar and intuitive for a common use. For this reason, we can rely on this foundational ontology and borrow some concepts to either map or specialize into more elaborated concepts and relations. Indeed, in line with what proposed by Noy and McGuinness (2001), we opted for extending a well-founded communication ontology.

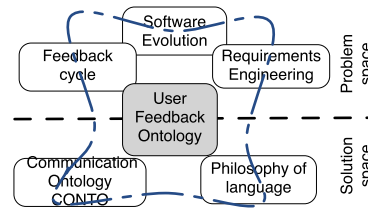


Fig. 1. Overview of the relevant background.

2.1. User feedback in software engineering

The analysis of user feedback is relevant for software evolution. Evolution may concern new designs evolving from old ones, as stated by Godfrey and German (2008). In addition, the evolution of software, according to Lehman (1980), is driven by human decision, which are taken in several stages of the software development. Moreover, the eighth law of Lehman's laws of software evolution, i.e. *Feedback system* is said to be the most subtle and complex because this law recognizes the role of user feedback as a need of change (Godfrey & German, 2008).

Madhavji et al. (2006) define feedback as "Feedback is one of the primary results of introducing the implemented software system into the real world. There is an immediate response to the system from those affected by it". This definition leads us to recognize explicit user feedback both as an artifact and as a process, i.e. a specific type of communication process. Taking the perspective of user feedback as an artifact, we revise the previous definition and propose the following one: "*User feedback is a reaction of the user upon her experience in using a software service or application. It could be based on multi-modal communication, such as natural language text, images, emoticons, etc. Moreover, it contains meaningful information with the purpose of suggesting improvements, i.e. requesting new needs, reporting failures, and asking for modifications or clarifications*". Taking the perspective of the communication process, we claim that the roles of sender and receiver are key to make clear the purpose of explicit user feedback.

Regarding the feedback in requirements discovery, Robertson indicates that "effective feedback is received when somebody understands something well enough to be able to constructively react to it" (Madhavji et al., 2006). In other words, this feedback should be received, analyzed, understood and put into practice by deriving requirements from it. But also she highlights that for accomplishing the goal of stimulating the reception of feedback it is necessary to provide appropriate mechanisms to collect it according to the stakeholder.

This leads us to the first two phases out of four phases of the *feedback cycle*, namely, *feedback collection* and *feedback analysis* (Brun et al., 2009).¹ Then, the cycle starts with the collection of data, in our case, coming from the users' experience who usually provide comments regarding the software at use. Next phase corresponds to the analysis of such comments, and there are approaches that apply natural language processing techniques for this purpose.

The speed of the feedback analysis sometimes depends on the way the feedback is collected. What we mean is that for example there are platforms that collect the feedback by letting the user classify it as feature request, bug report, enhancement, etc.² On the other side, there are approaches that analyze only one class of feedback, e.g. feature request (Cleland-Huang et al., 2009), or that analyze the feedback to determine the different classes such as bugs, features or clarifications (Carreño & Winbladh, 2013).

¹The other phases are *plan* and *act*.

²Example of this is the issue tracking system of OpenOffice <https://bz.apache.org/ooo/>.

2.2. Communication ontology

Collecting user feedback basically consists in a process of communication between the developers and the users of existing software systems. Thus, our work extends a Communication Ontology, developed by Oliveira et al. (2007), to include user feedback concepts. This ontology is specially attractive because it is grounded on a foundational ontology and also based on a well-known linguistic theory by Searle (1983). These choices contribute to make this a consistent and sound ontology of the communication field. The communication ontology was developed as an extension of the Unified Foundational Ontology (UFO) (Guizzardi, 2005; Guizzardi et al., 2008).

The development of domain ontologies by extending foundational ontologies that take truly ontological issues seriously is becoming more and more accepted by the ontology engineering community (Guizzardi et al., 2010). UFO has been successfully applied to provide real-world semantics for ontologies in different fields, such as Software Process (de Oliveira Bringuente et al., 2011), Petroleum (Guizzardi et al., 2010) and Collaboration (Oliveira et al., 2007). The Communication Ontology used in this work has been developed as a third component of the mentioned Collaboration Ontology. In the following two paragraphs, we discuss the concepts that we reuse from UFO and from the Communication Ontology.

UFO distinguishes between endurants and perdurants. Endurants do not have temporal parts, and persist in time while keeping their identity (e.g. a person and the color of an apple). A Perdurant (also referred to as event), conversely, is composed of temporal parts (e.g. storm, heart attack, trip). Substantials are existentially independent endurants (e.g. a person or a car). Moments, in contrast, are endurants that are existentially dependent on other endurants or events, inhering in these individuals (e.g. someone’s headache and the cost of a trip). Moments can be intrinsic or relational. Intrinsic Moments are those that depend on one single individual in which they inhere. A Relator is a relational moment, i.e. a moment that inheres simultaneously in multiple individuals. In Fig. 2 we present the concepts that we are borrowing in order to create our ontology. Agents are substantials that can perceive events, perform action contributions and bear special kinds of intrinsic moments, named Intentional Moments (examples of agents are person, student and software developer). Action contributions are intentional participations of agents within an event (e.g. saying something to someone, writing a letter). An Intention is a type of intentional moment (other examples of intentional moments are belief and desire) that represents an internal commitment of the agent to act towards that goal and, therefore, causes the agent to perform action contributions.

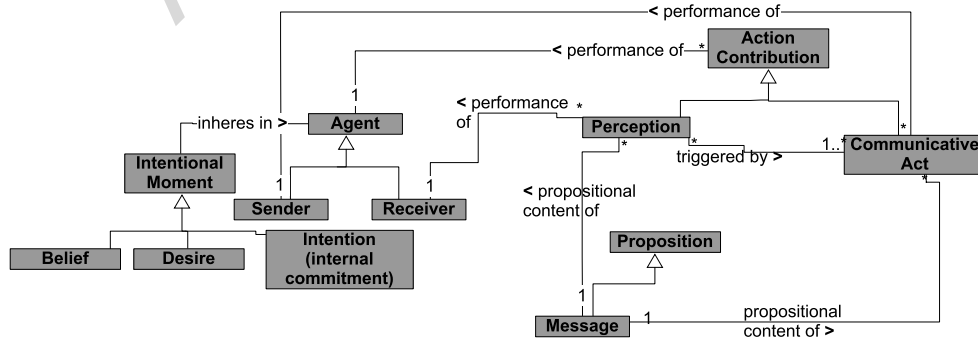


Fig. 2. Communication Ontology.

The communication ontology considers sender and receiver the two types of agents that are important in the communication process. The **Sender** is an agent that sends a message through a communicative act. A **Communicative Act** is an action contribution that carries out the information exchanged in the communication process. This communicative act is what Searle terms illocutionary act (Searle, 1983). The exchanged information is here captured as a **Message**, which is the propositional content of the communicative act. A **Receiver**, on the other hand, is an agent that perceives the communicated message. As the communicative act, a **Perception** is an action contribution that consists in the reception of the exchanged information (thus, also having a message as propositional content). A **Communicative Interaction** is a complex action composed of exactly one communicative act and one or more perceptions. In other words, in a communicative interaction, there is one sender agent and at least one receiver agent. Moreover, the communicative act and the perceptions involved in a communicative interaction have the same message as propositional content. Thus, this message is also said to be the propositional content of the communicative interaction.

2.3. Philosophy of language

In philosophy of language several studies have focused their attention on the intentionality that is conveyed through messages. Alessandro Duranti (2006) says that “there exists a level of intentionality that is distinct from the particular conceptualizations offered by a particular language or discourse”. Grice’s inferential model of communication states “a communicator provides evidence of her intention to convey a certain meaning, which is inferred by the audience on the basis of the evidence provided” (Wilson & Sperber, 2002). Moreover, Austin (1962) and Searle (1969) with the Speech Act Theory (SAT), claim that when a person says something she/he attempts to communicate certain things to the addressee, which affect either their beliefs and/or their behavior. By considering this concept of intention we found that is the speech act theory which treats the utterance of people as intentions to be interpreted. This is detailed in the following lines.

2.3.1. Speech act theory

The Speech Act Theory (SAT) was initiated and developed by Austin and Searle, respectively, in the field of philosophy of language (Austin, 1962; Searle, 1969). SAT claims that when a person says something she/he attempts to communicate certain things to the addressee, which affect either their beliefs and/or their behavior. A speech act involves three types of acts, namely, locutionary, illocutionary and perlocutionary acts. A locutionary act is the act of “saying something”, an illocutionary act makes reference to the way in which the locutions are used and in which sense, and a perlocutionary act is the effect on the audience that may be achieved. So for instance, considering the utterance “I’ll bring you a chocolate”, the locutionary act corresponds to the utterance of this sentence, the illocutionary act corresponds to the speaker’s intention to make the audience aware that she is committing to bring a chocolate, and the effect, i.e. the perlocutionary act, is that the audience got convinced about the speaker’s intention. It is important to underline that sentences contain the so called performative verbs. These verbs describe an action or intention to be performed, therefore they *classify* the illocutionary act into different categories proposed initial by Searle. From the previous example, “I’ll bring you a chocolate”, “I’ll bring” expresses my intention (technically named *illocutionary act*) to make you aware that I’m committing to bring a chocolate, and the effect (technically named *perlocutionary act*), is that you got convinced about my intention and expect to receive a chocolate. According to the extended classification of speech acts proposed by Bach and Harnish (1979), this type of *speech-act* is called *Commissives* since it expresses the speaker’s intention to commit to do something for the benefit of the hearer. Other types are: the

Constatives, which express the speaker's belief and her intention or desire that the hearer forms a like belief; the *Directives*, which express the speaker's attitude toward some prospective action that should be performed by the hearer; and the *Acknowledgements* that express the speaker's intention to satisfies a social expectation. There are four main kinds of acts, namely constatives, directives, commissives, and acknowledgements.

- (1) *Constantives* express the speaker's belief and her intention or desire that the hearer has or forms a like belief, e.g. "I must confess I'm a good chef".
- (2) *Directives* express the speaker's attitude toward some prospective action that should be performed by the hearer and her intention that her utterance must be taken as a reason for the hearer's action. For example, if I say to you: "I want you to walk the dog in the evening", I intend to motivate you to perform the action.
- (3) *Commissives* express the speaker's intention to commit to do something. As for instance when I say to you: "I am going to bring you a chocolate", I intend to make you believe that I'm committing to buy and bring a chocolate for you.
- (4) *Acknowledgements* express feelings regarding the hearer or the speaker's intention that her utterance satisfies a social expectation. For instance, "Please forgive me".

Here below we present in detail the different specializations of speech acts that are depicted in Fig. 3. We give the definition, taken and adapted from Bach and Harnish (1979), and the examples regarding each type.

A Constative speech act refers to an expression of a belief, together with the expression of an intention that a receiver form a similar belief. The specific kinds of constative acts are confirmatives, assertives, descriptives, suppositives, suggestives, informatives, responsives and concessives. Confirmatives acts express not only sender's belief about something but that she/he believes it as a result of some truth-seeking procedure, such as investigation, argumentation or observation.

Example 1. "Issac Newton formulated the laws of motion".

Assertives acts are considered as having a strong belief and intention by a sender who maintains his/her belief about something.

Example 2. "I know the chocolate is good for health because a scientific research has been developed on the topic".

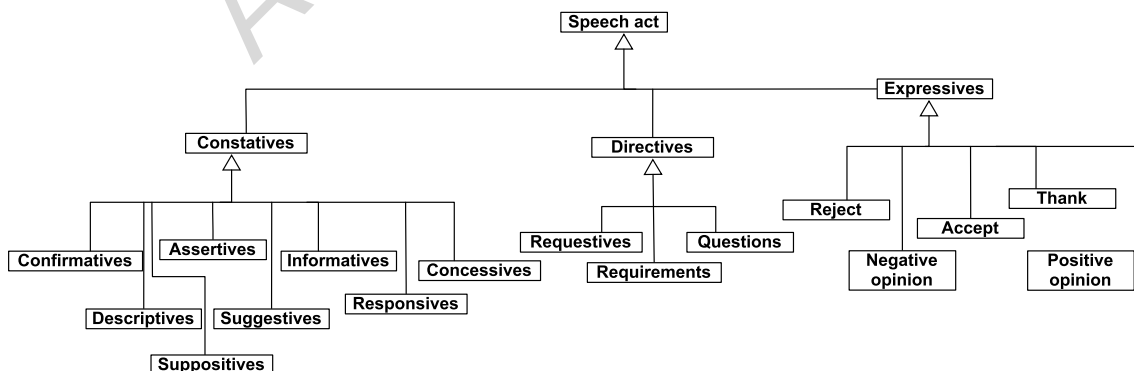


Fig. 3. Revised taxonomy of speech acts, adapted from Bach and Harnish (1979).

The Descriptives acts are used by a sender to make the receiver form a similar belief about something by given more details, then giving arguments to believe.

Example 3. “The Embassy building is painted white and orange, and it has many windows”.

Suggestives and Suppositives express only the belief that there is a reason to believe something but not sufficient reason to believe, and that is worth considering the consequences of something regardless of whether it is true, respectively.

Example 4. “I suggest you to try this Japanese restaurant, it has many positive comments”.

Example 5. “I suppose there is a train at midnight”.

The Informatives acts are considered as default acts used by a sender trying to inform a receiver about something that is presumed she or he does not know.

Example 6. “Rome is the most important tourist destination in Italy”.

Responsives are intended as replies from the receive to an inquiry by the sender.

Example 7. “Regarding your question about the best beer, to me, all German beers are good, hope this helps”.

And Concessives express a belief contrary to what the sender would like to believe or contrary to what he/she previously believed.

Example 8. “I must confess that I couldn’t finish the task”.

Directives express the sender’s attitude toward some prospective action by the receiver. We have selected only three subkinds of acts in this category: requestives, requirements and questions. Requestives express sender’s intention that the receiver take the expressed desire as reason to act.

Example 9. “I kindly ask you to provide me some info about any good library to work with graphs”.

Questions are special type of requests due to what it is requested is that the receiver provide the sender with certain information, but with the syntax of a question.

Example 10. “Can anybody explain me how to configure the server ‘X’?”

And Requirements acts are stronger than Requestives, since the sender’s belief is that his/her utterance constitutes sufficient reason for the receiver to perform an action.

Example 11. “I want to have a toolbar with effects to apply to photos”.

The last main category of speech acts is Expressives that refers to feelings expressed by a sender. Within this main category we have revised some acts that we consider pertinent for our research purpose. These acts are reject, accept, thank, negative opinion and positive opinion. Reject and Accept acts refer to one’s explicit attitude with respect to some statements.

Example 12. “I totally disagree on having a new color palette”.

Example 13. “I agree on the idea of changing the interface”.

While Negative opinion and Positive opinion acts reflect the appreciation that the sender may have regarding something.

Example 14. “I don’t like the new check spelling function”.

Example 15. “The new option to export video is very good”.

We have to point out that these two previous concepts are also aligned with the work of SentiWordNet in opinion mining (Esuli & Sebastiani, 2006).

3. User feedback ontology

In this section, we first present the conceptual map we built through a literature review,³ which provided the input for building the goal-oriented model of the user-feedback context and helped us capture the competency questions to be answered by our ontology (Guizzardi et al., 2014). This conceptual map represented the basis towards the formalization of it as a well-founded ontology defined in terms of an OntoUML model.

3.1. Concepts from the literature

The Fig. 4 gives an overview of the main concepts of the user feedback context that were identified through the literature review. The concept software user indicates a person who provides feedback (in our case called feedback provider) in an explicit way upon using a software service or application. Explicit user feedback refers to the deliberated desire of commenting about a software application or service upon using it. The format of feedback that we are mainly interested in characterizing is the textual format. There may be, however, other presentation formats such as visual and audio, which are sometimes accompanying the textual one. Although implicit feedback is not focus of our attention it is

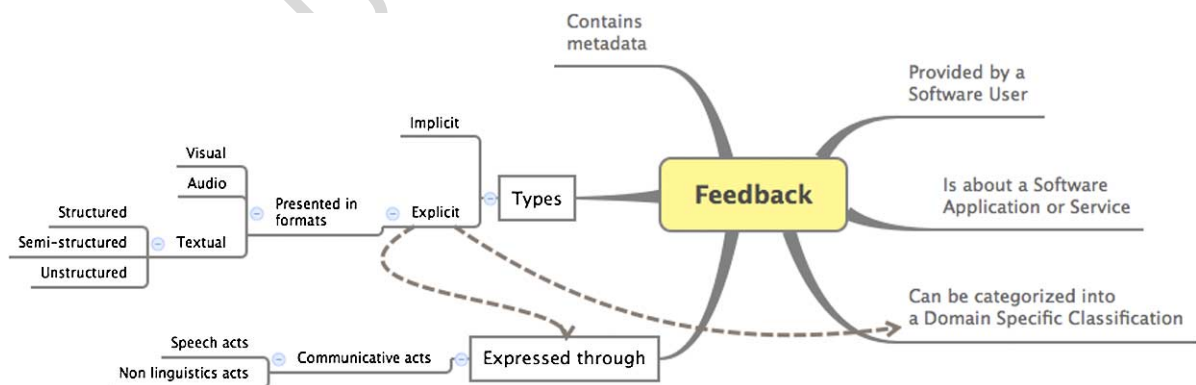


Fig. 4. Conceptual map of the user feedback domain.

³The literature review is part of the thesis of Morales-Ramirez (2015) available upon request.

worthy to mention that this type of feedback refers to the observable behaviors exhibited by a user, e.g., number of times listening a song or time spent surfing on a Web page (Jawaheer et al., 2014).

What is expressed through the explicit feedback is considered a communicative act that must be decoded, e.g., by reading or listening. Words and phrases must be interpreted and they may lead to discover speech acts and/or non-linguistic acts. A non-linguistic act can be interpreted by associating an intentionality as for speech acts. Hence, the specific non-linguistic act *gesture*, for instance “bowing”, can be interpreted as a greeting or as a thank. *Drawing* is another non-linguistic act that refers to emotions revealing a sender’s intention. And focusing on online discussions, we can consider information contained in attached documents as another specific non-linguistic act. For example, in the context of software users forum we can interpret as non-linguistic act the inclusion of excerpts of code, log files and URL links that a sender may attach to messages that she might write to participate to an online discussion. The ultimate purpose of providing explicit feedback by users of these forums, is to notify that a software application or service in use has a bug, needs to be improved, is missing something or that certain functionalities are not well understood. Indeed, in the domain of software applications, feedback is often gathered through platforms that support the formulation of structured or semi-structured messages, e.g. OpenOffice Bugzilla platform. Analogously, in other marketing domains, different contact forms are made available to customers, taking into account different needs (intention) the customer might have, such as getting technical help on a specific service, or providing information updates for billing. This finding from the literature review, is captured in the conceptual map by the concept of Domain Specific Classification, which is connected to the concept of explicit feedback. The main problem of the described conceptual map is that it does not rely on an exhaustive analysis of concepts and their relations. The relations are not properly assigned and the use of terms for representing a concept have not been ontologically founded, i.e., the concepts have not been thought from their essence or nature of being. As previously said, one advantage of the ontologies is being a complete and coherent model of a particular domain that help compare and more generally enable an inter-operation of different techniques, as for instance feedback gathering and analysis techniques. Another important benefit expected from the user feedback ontology, opportunely combined with a specific domain ontology, is to use it for understanding what the users are expressing. So for instance, in the domain of software applications, it would help developers in closing the gap between the current state of a software application and the desired performance by the users.

3.2. *OntoUML specification of the ontology*

Figure 5 shows the extended communication ontology, already introduced in the background section, but modeled in OntoUML (Guizzardi, 2005).

In the previous version of the ontology we maintained compatibility with the communication ontology (Oliveira et al., 2007), which was based on UFO-C. However, we opted to model the current version using OntoUML, which is based on UFO-A. This, along with some extensions, justifies the changes in the present ontology when compared to the previous version. Note that we took the decision of using OntoUML mainly to facilitate the evaluation (please refer to Section 5). An example of this can be observed when specifying that a *Sender* whose class is a *RoleMixin* and it is characterized by a *CommunicativeAct*. The communicative act is an intrinsic property belonging to the sender and that is why it is modeled as a type mode. For having a *Communication* it is essential that a communicative act is present. Therefore, the derived relation between the sender and the communication is present. In order to avoid cyclic relations, which could lead to unintended instances, a kind of anti pattern, we have added some OCL constraints to ensure that the relations are needed but one relation (sender.communication)

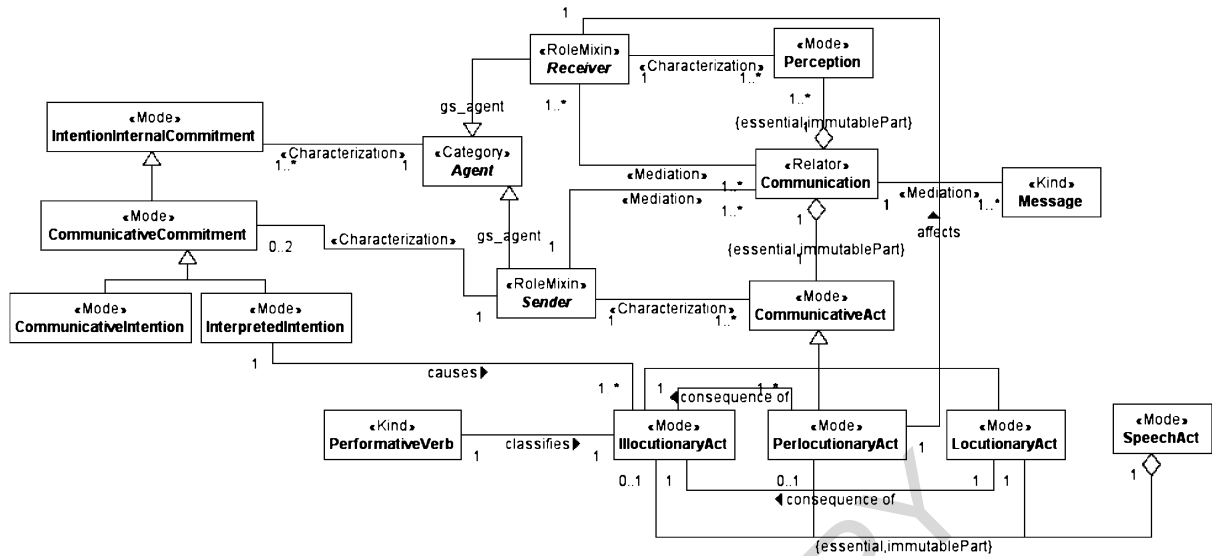


Fig. 5. Excerpt of the ontology describing the concepts extending the communication ontology, OntoUML notation (Guizzardi, 2005).

Table 1

OCL constraint to avoid the cycles between the relations sender-communication and receiver-communication

```

context _'Communication'::_'sender':_ 'Sender'
derive: self._'communications'._ 'communicative'._ 'sender'
context _'Communication'::_'receiver':Set(_'Receiver')
derive: self._'communication'._ 'perception'._ 'receiver'
    
```

is derived from the relation sender.communicativeAct. We can see the OCL constraints for the relations between sender and communication and receiver and communication in Table 1.

With the first OCL constraint, we want to make sure that, for each Communication instance (refer to the relator in Fig. 5), the Sender in the Communication is the same one related to the CommunicativeAct which is part of such Communication. Analogously, we add a similar OCL constraint to relate the Receiver with the Communication and Perception.

We have included some concepts in order to represent user feedback as textual comments containing speech acts. Specifically, two main original concepts were extended, namely, CommunicativeAct that is modeled as a mode entity as well as the mode IntentionInternalCommitment (referred as Intentional moment in the original ontology, see Fig. 2), which is an intrinsic moment of the agent.

The IntentionInternalCommitment is an intrinsic moment that any agent can commit to. In the specific case of a communication we have recognized that there is a CommunicationCommitment that is specifically achieved by a Sender. Indeed, this communicative commitment is specialized into CommunicativeIntention and InterpretedIntention. The CommunicativeIntention is a mode that refers to the internal commitment of a sender to convey an information to a receiver or an audience regardless of having this information understood by the audience. Differently, the InterpretedIntention is a mode that represents the formulation of a message by a sender with the aim of being recognized by a receiver and affecting her behavior. For example, if an expert in software system security is invited to give a talk to an audi-

ence of professionals with different backgrounds, s/he might have the only purpose of communicating information, without necessarily having in mind the purpose of such an information being understood by everybody. Vice versa, if the expert is invited to give a course in software system security, s/he has the commitment of making her way of speaking and explaining understandable by all the attendees, by recurring to special tools, materials or speech.

From the linguistic point of view, a speech act *involves* three acts, namely, locutionary, illocutionary and perlocutionary act that we consider specializations of a communicative act. Figure 5 shows the refinement of the relation between speech act and communicative act, by connecting the speech act entity directly to the modes LocutionaryAct, PerlocutionaryAct, and IllocutionaryAct, those entities represent the corresponding concepts proposed in SAT. LocutionaryAct is the act of “saying something” (production of words), an IllocutionaryAct makes reference to the way in which the locutions are used and in which sense (intention that motivates the production of words), and a PerlocutionaryAct is the effect on the receiver or audience that the sender wants to be accomplished.

For instance, considering the utterance “*Is there any example code I could look at?*”, the locutionary act corresponds to the utterance of this sentence, the illocutionary act corresponds to the speaker’s intention to make the audience aware that she has a request, and the effect, i.e. the perlocutionary act, is that the speaker got the audience to handle her request. A speech act involves at least one act, meaning that someone could utter a sentence that does not make any sense, for instance “one the snow”, accomplishing the locutionary act, but the illocutionary and perlocutionary acts are not present. We can observe in the model how the speech act is composed by all the acts but only the locutionary act is mandatory. Besides this, if the illocutionary act is present, consequently there can be a locutionary act and one or more perlocutionary acts.

A sender, being an agent that has intentions, in the context of feedback the sender has the specific communicative commitment that is the interpreted intention, which *causes* the illocutionary act. Therefore, the locutionary act is a *consequence* of this illocutionary act through the utterance of specific words that will reify such a interpreted intention hopefully successfully perceived by a receiver. Another *consequence* of the illocutionary act is the perlocutionary act that the receiver will perceive and might be affected by the speech act. We need to clarify that the relation *consequence of* between the illocutionary and perlocutionary act is a type of indirect causation, i.e. if the communication is successful, the receiver will perform the action intended by the sender.

The PerformativeVerb is a kind entity referring to the verb that *classifies* the illocutionary act into five categories defined by Searle (1969), later revised by Bach and Harnish (1979). Example of performative verbs are: confess, agree, suggest, disagree, support, suppose, etc. Finally, a speech act is successful if the intention that the sender expresses is identified by the receiver by means of recognizing such a sender’s intention. The concept speech act is extended with the addition of the NonLinguisticAct that were previously introduced. In Fig. 6 we can observe that the non-linguistic acts cannot be considered as a specialization of a speech act but they can be interpreted in terms of the categories of speech acts.

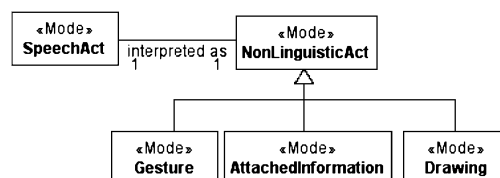


Fig. 6. Non-linguistic acts extending the speech acts taxonomy.

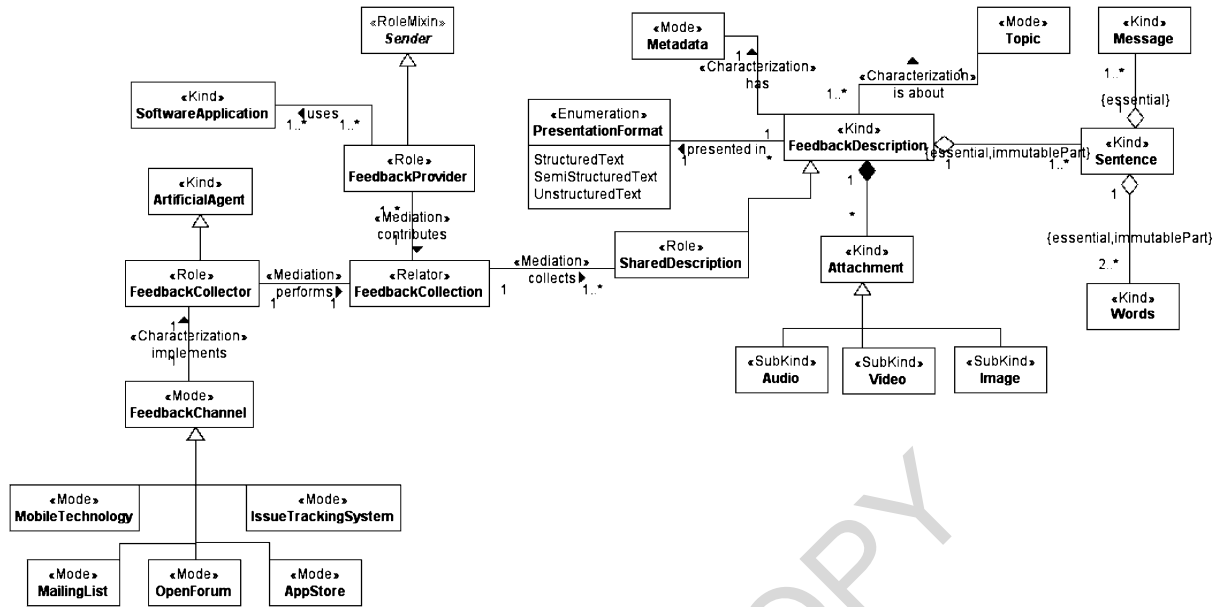


Fig. 7. Excerpt of the ontology describing the user feedback elaboration, provided by the stakeholder.

The extension to the communication ontology gives us the starting point for developing the two perspectives of the user feedback ontology. The first perspective corresponds to the software user who by adopting the role of a sender enables the elaboration of user feedback. In our ontology we call her/him feedback provider.

In the following, we describe the concepts belonging to this perspective, which is depicted in Fig. 7. The role *FeedbackProvider* is assumed by a *Sender* when s/he contributes to the *FeedbackCollection* through providing a *SharedDescription*, which is a specialization of the *FeedbackDescription*. The role of *FeedbackCollector* can be performed by an *ArtificialAgent* whose concrete instantiation becomes a *FeedbackChannel* (a mode) specialized into different platforms, such as *MobileTechnology*, *MailingList*, *OpenForum*, *AppStore* or *IssueTrackingSystem*.

The kind entity *FeedbackDescription* is composed of at least one *Sentence*, which is composed of messages, and each *Message* represents the *propositional content* of the Communicative act which is formulated in terms of *SpeechAct*, depicted in Fig. 6. The sentence is composed of *Words* that contribute to reify the speech act, indeed, a performative verb is a special type of word. The concept *Topic* is a mode (characterizing the feedback description) whose aim is to keep the messages in a coherent context of a discussion that gives an idea of what the feedback description is about (Blakemore, 2003). Two other important concepts related to the feedback description are the *Metadata* and *Attachment*. The former is modeled as a mode entity characterizing additional information such as the date of elaboration, IP address or any other important information regarding the elaboration of the feedback description. The attachment refers to a kind entity which is a non-linguistic act, previously explained, that a feedback description may contain or not, thus being also considered part of the user feedback. The specializations of the concept attachment are the subkind *Image* (e.g. emoticons), *Video* (e.g. recording a software flaw), and *Audio* (any voice-recorded message, e.g. in the *iRequire* system proposed by Seyff et al. (2010)).

Feedback description can be presented in different *PresentationFormat*, such as *StructuredText*, if specific fields are requested, e.g. select options and text boxes, *UnstructuredText*, that is free text or

SemiStructuredText that is combining the structured and unstructured formats as proposed by Schneider (2011).

The other perspective of the ontology corresponding to the analysis of user feedback is shown in Fig. 8. A person can assume the role of FeedbackAnalyst when is the Receiver of the feedback description and carries out a FeedbackAnalysis. The feedback analysis is a relator that makes possible that an analyst applies an AnalysisTechnique, e.g. speech acts analysis, sentiment analysis (depicted as types of AnalysisTechnique).

Any feedback analysis *generates* FeedbackAnalyzed that is the role that any feedback description assumes once it has been analyzed. Once the feedback is analyzed according to any analysis technique, a FeedbackClassification (which is a relator) should be performed in order to assign a category or type of feedback. The feedback classification that may correspond to a model of classification, practically uses the words as input of any probabilistic model, in its process of assignment of categories. In Table 2 we specify in OCL that the *«Mediation»* relation *uses* is a derived relation of the *«Mediation»* relation *classifies*, again to avoid a cycle.

Specific feedback types that can be assigned during the feedback classification are defined depending on the domain at interest. Since we focus on software applications domain, the analyzed feedback can

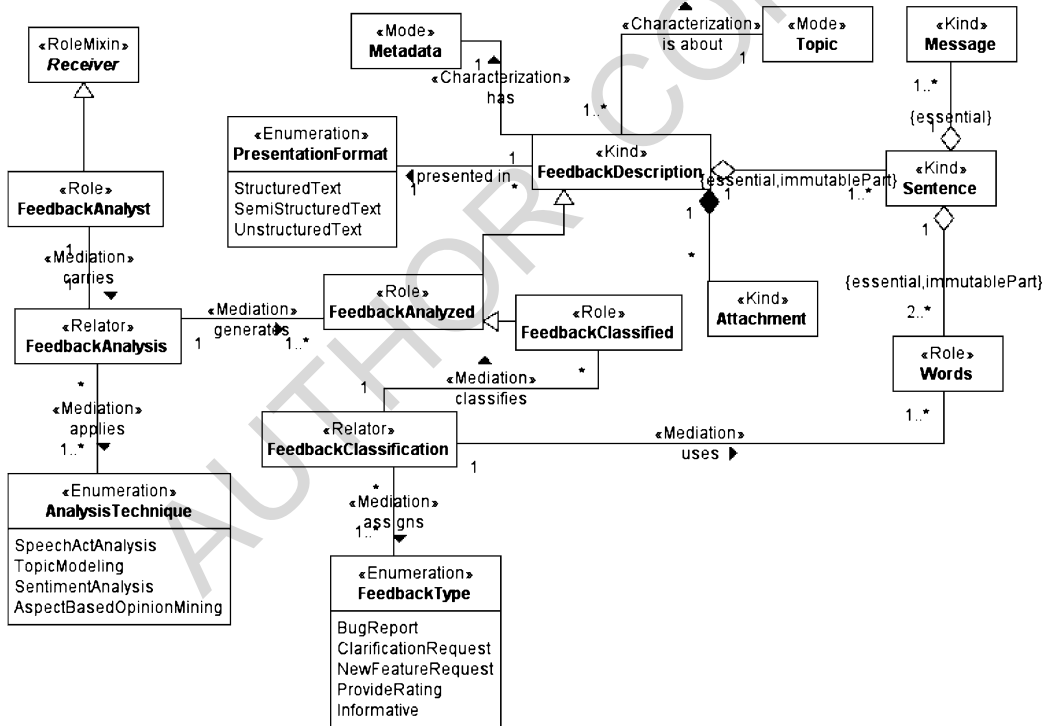


Fig. 8. Excerpt of the ontology describing the role of the feedback analyst.

Table 2

OCL constraint to avoid the cycle between the concepts feedback classification-feedback analyzed-words

context 'FeedbackClassification'::_'words':Set(_'Words')

derive: self._'feedbackclassification'._'feedbackanalyzed'.oclAsType(_'FeedbackDescription')._'sentence'._'words'

be classified into different types according to the software user's interpreted intention (thus changing the feedback description from the role `FeedbackAnalyzed` to the role `FeedbackClassified`). Possible intentions for a software user to send a feedback are listed in the enumeration `FeedbackType` such as: `ClarificationRequest`, `NewFeatureRequest`, `BugReport`, `ProvideRating` or simply `Informative`. We chose these types to be the usual interpreted intentions based on the literature of user feedback or because they are widely used in practice, for instance in issue tracking systems.⁴

For instance, `Bug Report` is inspired from *defect*, *corrective* or *negative* feedback (Hattie & Timperley, 2007; Brun et al., 2009). If the user feedback refers to information that should be considered for performing a correction of the software, this means that such an information has a negative connotation. The concept of feedback for *encouraging* as discussed by Mory (2004), and that of *positive* feedback referred by Brun et al. (2009) have been captured in `ProvideRating`, according to the terminology proposed by Pagano and Maalej (2013). Analogously, concepts expressed with terms like *strategic behavior* and *enhancement* type (Mory, 2004), are referred as `NewFeatureRequest` in our ontology. While `ClarificationRequest` represents the case when the feedback contains questions or extra information (such as critical details) to make something clearer. The `Informative` type can be used in the case that none of the previous types satisfy an interpretation.

As highlighted in the above description, the concepts in our feedback ontology are indeed used in different works, but often expressed with other terms and synonyms. The proposed ontology attempts at providing a way to unify all the different concepts in a single classification.

4. Application of the ontology

So far, we explained the concepts of the proposed ontology with the help of simple examples. We consider now four scenarios taken from the software engineering context, and use them to provide a concept-by-concept walkthrough of the proposed ontology. The main problem at interest in the considered scenarios is the analysis of the user feedback for the purpose of identifying new features or requirements to be implemented in next software releases. The first three scenarios highlight the relationship between the feedback channel used for gathering the user feedback and the analysis that is performed manually by the requirements engineer. The fourth scenario refers to an approach for a semi-automated analysis of online user feedback in the context of Open-Source Software (OSS) development.

4.1. Scenario 1: Unstructured user feedback

The mailing list is a feedback channel that allows stakeholders to communicate user feedback in an unstructured format. This format gives more freedom to the stakeholders for expressing their needs or complains, but on the other hand, it leads to a time-consuming task for developers to process it. Next, we describe how manual analysis of unstructured feedback is performed by using concepts of the proposed ontology, concepts depicted in Fig. 7, and in Fig. 5. The `FeedbackChannel` selected is the mailing list, which allows the development of discussions where stakeholders interact and discuss about a `SoftwareApplication` called `XWiki`.⁵ We randomly selected two `FeedbackDescriptions` (i.e., the first e-mail of two different threads), we show the first example in Table 3.

⁴<https://bz.apache.org/ooo/>.

⁵<http://www.xwiki.org/xwiki/bin/view/Main/WebHome>.

Table 3
Example 1: unstructured feedback

Subject: <u>Latex html in editor for math formulas?</u>	<i>Topic</i>
From: <u>alan</u>	<i>Sender</i>
Body:	
...	
1: <u>Will there be an additional button in the editor, to implement formulas?</u>	<i>Question</i>
2: <u>Or is there?</u>	<i>Question</i>
...	
3: <u>Or how can I use formulas, maybe you know a good latex html live editor.</u>	<i>Suppositive</i>
4: <u>Since xwiki is more and more used by universities and labs...</u>	<i>Informative</i>
5: <u>it would be nice to have something,</u>	<i>Requirement</i>
6: <u>if there already isnt?</u>	<i>Question</i>
7: <u>Thanx,</u>	<i>Thank</i>
8: alan	

E-mails have Metadata containing information such as the sender of the e-mail, date, the subject of the e-mail, among other information. For us is important to identify what is the topic, who is the stakeholder sending the e-mail and the feedback description expressed in the e-mail. As illustrated in Table 3,⁶ the Topic is given in the field –Subject:– of the e-mail, the stakeholder, who we call FeedbackProvider thus the sender is given in the field –From:–, and the FeedbackDescription is given in the section –Body:–.

The Topic underlines the context in which the FeedbackDescription must be interpreted.

At this point, we connect the part corresponding to the extended communication ontology and the user feedback elaboration by the elements Sentence and Message (see Figs 5 and 7). As we already mentioned the FeedbackDescription is composed of Sentences that are enumerated in the running example. Using as support the explanation of each specialization of SpeechActs (see Section 2.3), we discover that each sentence of the FeedbackDescription contains Messages that are an essential part of the communication and can be interpreted with a certain intention (i.e., InterpretedIntention). For example, the sentence of line #1 is a CommunicativeAct expressed by the Sender –alan– who desires that his InterpretedIntention (made by the developers of XWiki) would be a Question, whose PerlocutionaryAct would be the expectation of an answer by the Receiver –someone from the XWiki community–.

The sentence of the line #2 is also a Question, in line #3 there is a Suppositive sentence, line #4 refers only to an Informative act, while sentences or line #5 and #6 are a Requirement and Question, respectively. The last lines #7 and #8 are a thank and the name of the stakeholder/sender.

⁶Source: <http://lists.xwiki.org/pipermail/users/2008-February/006334.html>.

Now, we look at the excerpt of the ontology describing the feedback analyst (see Fig. 8) and we can recognize that what we have done until now is a *FeedbackAnalysis* of the description applying the *AnalysisTechnique SpeechActAnalysis*. Therefore, the splitting of the description into sentences and the annotation of speech acts makes of the feedback description to become *FeedbackAnalyzed*. What is important to highlight is that so far the ontology support us manually to characterize the e-mail's body from a linguistic and intentional point of view (i.e., the *AnalysisTechnique –speech acts–*), but it's up to the *Receiver* (i.e., a possible developer from the XWiki community) who has to reflect about all the set of speech acts and their intentions, and classify the *FeedbackAnalyzed* according to the *FeedbackTypes* we provide for the *FeedbackClassification*. Since the presence of an explicit *Requirement* is expressed along with several *Questions*, we could say that this example of *FeedbackDescription* is a candidate for being handled as a *NewFeatureRequest*.

The second example in Table 4⁷ has also the instances of the *Topic* and the *Sender*. In the *FeedbackDescription* we find that the sentences of line #1 and #3 are *Informative*, but line #3 also contains the speech act *PositiveOpinion*. While sentences of line #2 and #4 can be interpreted as *NegativeOpinion*. The last sentences of the lines #5, #6 express the corresponding specializations of the speech acts *Request* and *Thank*.

The last line #7 corresponds to the name of the sender –Niko–. Again, we only want to make clear that the *InterpretedIntentions* that we are defining are founded on the baseline of our work, regarding the communication ontology and the SAT.

The *Receiver*, in charge of evaluating how the feedback can be classified and thus giving a follow up, can consider to handle this *FeedbackDescription* as a candidate to be *FeedbackClassified* as *BugReport*,

Table 4
Example 2: unstructured feedback

Subject: Horizontal Menu like xwiki.org in Farm
Topic

From: niko
Sender

Body:

...

1: i just created my own farm and try to build a template wiki which i can use...
Informative

2: now i got some problems with the configuration.
Negative opinion

...

3: if i go to the main wiki i can use this menu and it works...
Informative *Positive opinion*

4: but on my subwiki it wont work..
Negative opinion

5: hopefully someone is more experienced in running a farm and culd help me out.
Request

6: thanks so far and regards
Thank

7: Niko

⁷Source: <http://lists.xwiki.org/pipermail/users/2012-September/023787.html>.

because there are two indicators of problems, lines #2 and #4, as well as the request for help expressed in line #5.

4.2. Scenario 2: Semi-structured user feedback

The issue tracking system is a `FeedbackChannel` that allows stakeholders to communicate user feedback in a semi-structured format. It means that there are some mandatory fields to be filled in together with the `FeedbackDescription`, which is freely written. The extra fields to be captured represent the extra information that can be contained in the `Metadata` and an initial `FeedbackClassification` given by the stakeholder, who provides the feedback.

The semi-structured feedback is usually provided through the `FeedbackChannels` called issue tracking systems, such as the Apache OpenOffice Bugzilla platform. The `SoftwareApplication` is OpenOffice⁸ and we have selected two `FeedbackDescriptions` (i.e., the first comment of two different discussions already classified). Since the semi-structured feedback is already classified, we can use this as an advantage to understand how the ontology concepts that we can find in the semi-structured feedback, i.e., the different specializations of `SpeechActs` are usually expressed according to the assigned `FeedbackClassification`.

In Table 5⁹ we present a first example, the topic can be identified after the number of the issue –Issue 11121–. The `FeedbackClassification` is identified by the –Issue type:– and the `Sender` is identified with the field –User:–, in this case is –Unknown–. As it can be observed, the type of classification used is –Enhancement– which is similar to the type –new feature request–, which is defined in our ontology as unification of alike words. With the use of the ontology we can add –Enhancement– as a synonym and create a domain specific synset for the types of `FeedbackClassification`, in the context of user feedback analysis for software evolution. The `FeedbackDescription` is given in the section –Description:–, wherein we can find that there are specialized `SpeechActs` `Suggestive` in line #1 and the `NegativeOpinion` in line #2. The last two sentences are only `Informative` speech acts. Something interesting to pinpoint is the

Table 5
Example 1: semi-structured feedback

Issue 11121	<u>Styles for tables</u>	<i>Topic</i>
Issue type:	<u>Enhancement</u>	<i>Feedback classification</i>
User:	<u>unknown</u>	<i>Sender</i>
Description:		
1:	<u>It's suggested to create styles for tables as for paragraphs, characters, numbering, frames and pages</u>	
	<i>Suggestive</i>	
2:	<u>Auto formatting does not allow to modify table properties well.</u>	
	<i>Negative opinion</i>	
3:	<u>For example, paragraph style for heading and contents, . . . should be configurable in style.</u>	
	<i>Informative</i>	
4:	<u>The reason is to simplify keeping of a number of documents in one style.</u>	
	<i>Informative</i>	

⁸<http://www.openoffice.org/>.

⁹Source: https://bz.apache.org/ooo/show_bug.cgi?id=11121.

Table 6
Example 2: semi-structured feedback

Issue 125096	Corrupt display of documents while scrolling
	<i>Topic</i>
Issue type:	Defect
	<i>Feedback classification</i>
User:	Thorsten Wagner
	<i>Sender</i>
Description:	
1:	Created attachment 83551 [details]
	<i>Attached information</i>
...	
2:	While scrolling display of documents becomes corrupt. The problem appears...
	<i>Negative opinion</i>
3:	Steps to reproduce:
	<i>Descriptive</i>
4:	(1) Open the sample document attached
	<i>Informative</i>
...	
5:	(3) Scroll up and down to the page break again... the footer is displayed correctly...
	<i>Positive opinion</i>
6:	The problem seems to disappear when removing images from header.
	<i>Negative opinion</i>
7:	The problem appears on Mac OS X 10.9.3... , but seems to exist in earlier revisions too.
	<i>Negative opinion</i>

expression of a negative sentiment in the description, but it may be due to the emphasis of the lack of such a functionality asked as an enhancement.

The second example of semi-structured feedback is shown in Table 6.¹⁰ The type of classification used is –Defect– which has been addressed as bug report in our ontology as a synonym. The Sender in this example is identified with a name. In the FeedbackDescription we find that the NonLinguisticAct appears as the specialized act attached information in line #1. We find also three sentences with a NegativeOpinion in lines #2, #6 and #7. And other speech acts such as Descriptive in line #3, Informative in line #4 and PositiveOpinion in line #5.

As previously said, this second example has been classified as a –Defect– and it is interesting to observe that the negative opinion is predominant in such a kind of feedback classification. Also worthy to mention is that in this case we see that an attachment has been added by the stakeholder as well as the description of steps to reproduce the error. Our ontology so far is not fully developed to support such a type of inference, but it provides hints about many possible characterizations of user feedback and the relations between concepts adds clarity for their interpretations on real instances. The usage of the ontology when analyzing semi-structured feedback can be beneficial in order to discover patterns of combination of specialized speech acts that are associated with a specific type of feedback classification. Therefore, we could evolve our ontology and incorporate these findings as relations between concepts.

¹⁰Source: https://bz.apache.org/ooo/show_bug.cgi?id=125096.

4.3. Scenario 3: Structured user feedback

This scenario shows the FeedbackDescription that is directly provided by the stakeholder –Sender– to the developer –Receiver–. Mobile applications or app stores are the common FeedbackChannels allowing stakeholders to directly communicate the user feedback in a structured format. By structured we mean that there is a specific input form where the free writing is significantly reduced and the users are guided step by step. Here it is also possible to add an image of the app, highlighting the specific functionality the textual description refers to. Specifically, we can say that there is a clear alignment between the feedback description and an image describing the problem or need, as we will see in the following two examples. The SoftwareApplication is the operating system (OS) of a smartphone, while the FeedbackChannel is a mobile App called AppEcho (Seyff et al., 2014).


We selected two examples of structured feedback and the first one is presented in Table 7.¹¹ The FeedbackDescription has been already classified as –Bug Report– that coincides with the one proposed in our FeedbackClassification. The Metadata accompanying this kind of feedback depends on the services provided by the smartphone, e.g., GPS localizer, date and time, Bluetooth, etc.

The SpeechActs found in this example are clearly NegativeOpinion both sentences. The data of the stakeholder –Sender– has been anonymized but the Receiver is the developer of the Bada platform.

In Table 8¹¹ we show the second example that has been classified as –Need– and it can be seen as synonym of –new feature request– of our FeedbackClassification. The types of specialized SpeechActs found are Requirement and NegativeOpinion. We again observe an interesting combination of SpeechActs that may confirm our idea that a Requirement made by a stakeholder can be emphasized with a NegativeOpinion to stress the lack of such a functionality. This also has been seen in Example 1 of the semi-structured feedback (see Table 5), wherein a Suggestive and a NegativeOpinion have been expressed by the Sender.

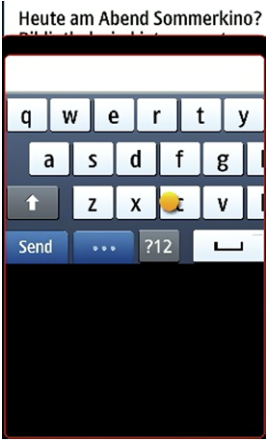
Table 7

Structured feedback accompanied with an image, classified as bug report

	<p>Time: 19:49 Feedback: <u>In the mail settings it is not possible... to enter...</u> <i>Negative opinion</i> ... the reason why my mail account is not working. <i>Negative opinion</i> GPS-Enabled: No. Anonymous Feedback Bada Platform: <u>Bug Report</u> <i>Feedback classification</i></p>
---	--

¹¹Source: Feedback collected through a mobile app, provided by Seyff et al. (2014).

Table 8
Structured feedback accompanied with an image, classified as a feature request

 <p>Heute am Abend Sommerkino? q w e r t y a s d f g ↑ z x ↵ Send ... ?12</p>	<p>Time: 08:40 Feedback: It would be nice, to have dot and comma on the... keyboard.</p> <p style="text-align: center;"><i>Requirement</i></p> <p>It is not possible to enter umlauts.</p> <p style="text-align: center;"><i>Negative opinion</i></p> <p>GPS-Enabled: Yes. Anonymous Feedback Bada Platform: <u>Need</u></p> <p style="text-align: center;"><i>Feedback classification</i></p>
--	--

4.4. Scenario 4: Semi-automated analysis of user feedback

OSS community members, for example OpenOffice users, report bugs, request features or clarifications by writing e-mails (mainly in unstructured natural language) to mailing lists. Requirements engineers examine them by reading huge threads of e-mails, so that automated support for retrieving relevant information and particularly for recognizing OpenOffice users' intentions (e.g., suggesting, complaining) can support analysts and allow them to increase the performance of this task. We propose a tool-supported method for the analysis of OSS mailing-list discussions in terms of the linguistic and non-linguistic acts expressed by the participants (Morales-Ramirez et al., 2014a). The method exploits NLP techniques for providing automatic speech-act classification, and an analysis model for this type of online discussion (i.e. OSS user forum), that helps requirements engineers to recognize indicators for bug and feature request descriptions.

With the help of the user feedback ontology, in the following we define this tool-supported analysis method, which is an instance of AnalysisTechnique. Specifically we describe the paradigm it builds on (i.e. SAT theory), and the analysis process it supports, including the preprocessing step for the characterization and collection of the tool's input.

- (1) *Speech-act based approach.* With the part of the ontology corresponding to the extended communication ontology (see Fig. 5) we defined the needed resources to be used in a tool for supporting an automated identification of speech acts expressed in messages of stakeholders.
 - (a) By taking as basis the taxonomy of speech acts from Fig. 3 and the proposed extension from Fig. 6 we defined a list of categories of speech acts that can be found in practice, i.e. in a real dataset of feedback descriptions.
 - (b) Elaboration of a list of PerformativeVerbs per each speech act defined in the previous list. In order to help the identification of the IllocutionaryActs caused by the Sender (i.e. user) in order to pursue that the Receiver make an InterpretedIntention of each CommunicativeAct expressed.
 - (c) Elaboration of lexico-syntactic rules for each speech act, using the list of performative verbs and surrounding words before and after it to create sequences of terms, called n-grams. These

n-grams are the instances of the specializations of speech acts, for example the speech act *Suppositive* has the instances (n-grams) such as “I guess”, “I suppose” or “I think”, among others.

- (2) *Characterization and collection of the tool’s input, i.e. feedback descriptions.* One important step is the characterization of the input to be processed by a tool, in order to make the best possible manipulation of it. With the support of the part of the ontology depicted in Fig. 7 we recognize that *FeedbackDescriptions* are composed of *Messages* and that these descriptions can be collected from different *FeedbackChannels*, which might imposed a type of *PresentationFormat*.
 - (a) Each *FeedbackDescription* must be cleaned of useless symbols or words, i.e. resembling html tags, signatures, stop words, etc., i.e. delete any noisy data no relevant for the understanding of the main messages.
 - (b) For practical purposes regarding the implementation, we characterize each sentence found in a feedback description as a *Message* that carries a propositional content, thus a speech act.
 - (c) The collection of *FeedbackDescription* is made by selecting *Topics*, which depending on the feedback channel can be identified differently in real datasets. For example, *FeedbackDescriptions* coming from mailing lists have their *Topic* in the *Subject* element of an e-mail, or the descriptions coming from App stores have their *Topic* in the element called “title” when reviewing a mobile application.
- (3) *Analysis process.* In this part we put into execution the tool using a real dataset of *FeedbackDescriptions*. The part of the ontology shown in Fig. 8 classifies our tool-supported method as an *AnalysisTechnique* called speech acts, which gives support to a *FeedbackAnalyst* in performing the *FeedbackAnalysis*.
 - (a) Once the messages are clean, each message is divided into sentences using a sentence parser.
 - (b) Each sentence is read by the tool and annotates the corresponding speech act according to the create rules.
 - (c) The *FeedbackClassification* must be manually performed by the *FeedbackAnalyst* based on the found speech acts and by considering one of the enumerations of *FeedbackType*.

5. Tool-supported validation of the ontology

Ontology validation activities consist in comparing the meaning of the ontology definitions against the intended model of the world that it aims to conceptualize (i.e., a frame of reference). That is, the question that the ontology validation attempts to answer can be expressed as: are you producing the right ontology?

We validated the proposed ontology by adapting the evaluation framework developed by Sabou and Fernandez (2012), and using tools such as the OntoUML Lightweight Editor (OLED),¹² which provides (among other validations) model checking functionalities through the integration of the Alloy model checker (Jackson, 2006). In this section, we introduce the overall evaluation framework and how we instantiated it. We describe also how eight iterations of the validation activity allowed us to produce an ontology with better quality. A discussion on the main findings from this validation process is also presented.

¹²<https://code.google.com/p/ontouml-lightweight-editor/>.

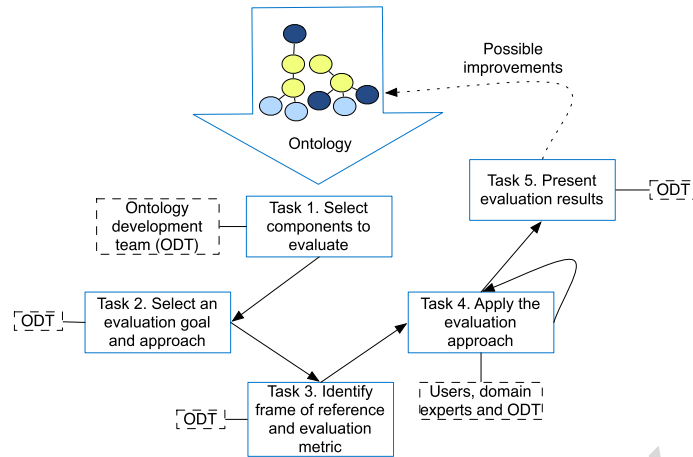


Fig. 9. Workflow and tasks to evaluate ontologies, adapted from Sabou and Fernandez (2012).

5.1. Evaluation workflow and its application

The evaluation framework adopted for validating the proposed ontology includes five main tasks, which are illustrated in Fig. 9. Notice that we focus on the evaluation of the ontology that extends the communication ontology, which includes the part describing the concepts of the elaboration of user feedback by a stakeholder, and the part aiming at supporting the analysis of user feedback. This required us to adapt the original workflow, which is meant for the evaluation of sets of ontologies (Sabou & Fernandez, 2012). In the following we describe each task, and how we performed it.

5.1.1. Defining the evaluation framework: Task 1–Task 3

Task 1. Select individual components to evaluate. This task is usually performed by the ontology development team (ODT). The ODT has to select one part of the ontology to be evaluated based on two criteria: (1) which elements of the ontology are critical for the ontology and (2) which of these elements can actually be evaluated. This means that a frame of reference must be defined in order to use it for evaluating the selected part of the ontology. A frame of reference is a perfect model to be compared to the object under evaluation, which is defined by a human evaluator. It is important to highlight that the frame of reference should not be necessarily tangible, but can be any kind of metric to resemble a perfect model defined by the evaluator for a particular evaluation task.

How we performed it. In line with our research purpose for the evaluation, we selected the concepts of the user feedback ontology that extends the foundational ontology, with the addition of three concepts that are originally defined in the communication ontology, namely *Sender*, *Receiver* and *Message*.

Task 2. Select an evaluation goal and approach. In this task the team must define the goal of the evaluation along with the selection of an appropriate evaluation approach. Here below, we present two examples of evaluation goals, taken from the set of goals presented by Sabou and Fernandez (2012):

- *Domain coverage* refers to the key question *does the ontology cover a topic domain?* The evaluation approaches guide the execution of the comparison of the ontology with respect to frames of references such as a gold standard ontology, or data sets that are representative for the domain (e.g., user-defined terms, tag sets, document corpus, etc.).

- *Quality of the modeling* in terms of the design and development process and in terms of the final result, driven by the question *does the ontology development process comply with ontology modeling best practices/ontology design patterns?* This evaluation goal focuses on the quality of the ontology which can be assessed using approaches focusing on logical correctness or syntactic, structural, and semantic quality.

How we performed it. We adopted as evaluation goal *Quality of the modeling*. Since we have modeled our ontologies with OntoUML, this means evaluating the quality of the OntoUML models we produce by exploiting available catalogues of anti-patterns. An anti-pattern has been defined by Guizzardi (2014) as “model structure that would systematically create deviations between the sets of valid and intended model instances”. Therefore, we aim at eliminating any found anti-pattern in our modeling, thus aiming at excluding any unintended instance and keeping the intended ones. This way, the quality of the developed ontology can be assured. Moreover, we can exploit the capability of a model-checker to instantiate the so called counter examples to identify unintended state of affairs that might be allowed by the ontology under evaluation.

Task 3. Identify frame of reference and evaluation metric. In this task the team decides which are the needed elements of the evaluation, consisting of:

- A *frame of reference* denotes the set of representative resources that sets a baseline value against which the ontology should be compared. Example of types of frames of references are:
 - * Gold standard: the frame of reference is defined by any kind of structured representation of the problem domain for which an appropriate ontology is needed. A gold standard is used when the goal of the evaluation is domain coverage.
 - * Assessment by humans: the frame of reference is defined by human judgements that measure ontology features.
- *Evaluation metric(s)* are specific for each frame of reference and can be for example precision and recall, measures of similarity between an ontology or a mapping, and a corpus (domain knowledge), and lexical terms. Example of evaluation metrics regarding the previous frames of references are:
 - * Metrics for gold standard: Interpretability: this metric is measured by checking WordNet to determine if the terms in an ontology are meaningful (i.e., existence of the individual words) (Burton-Jones et al., 2005). Clarity: this metric checks for the number of senses in WordNet for the class or property name as a whole, whether it is a single word or a phrase, (Burton-Jones et al., 2005).
 - * Metrics for the assessment of humans: Syntactic quality: this metric measures the number of syntactic errors found in the ontology (Burton-Jones et al., 2005). Essence: refers to a type of metric where the evaluator assesses if an entity is true in every possible world, according to Guarino and Welty (2009).

How we performed it. The frame of reference that we applied in our evaluation is a mix of *human assessment* and *tool-based assessment*. Indeed, we used the OntoUML Lightweight Editor (OLED) tool that allows ontology engineers to perform syntax verification, instance simulation by using Alloy, and evaluation of models for correcting anti-patterns, among other functionalities. The OLED tool lets ontology engineers transform automatically the OntoUML models into Alloy models to perform a model visual simulation checking. Moreover, as we aim at validating the essentiality of the existence of certain entities and properties in the real world, humans can perform such assessment on the visual instances produced by the tool.

We selected two metrics for the frame of reference. The first one is the so called *Syntactic quality metric*, which is defined in terms of number of anti-patterns found in the ontology. The recognition of anti-patterns is important because their inclusion in OntoUML models results into unintended model instances (i.e., representing undesired states of affairs). The second metric, called *Essence metric*, is measured by generating instances of the concepts and analyzing such instances to validate if there are over specifications or under specifications in the ontology that may lead to wrong relationships of concepts with respect to the real world.

5.1.2. Applying the evaluation framework: Task 4–Task 5

Task 4 refers to a proper setup for the designed evaluation (i.e. the result of Tasks 1–3), while Task 5 concerns the presentation of the results, which allows us to assess possible improvements along the evolution of the ontology.

How we performed it. We applied the evaluation framework by installing the OLED tool version 1.0.06 and by using it to model the ontology of user feedback with the OntoUML modeling language. Per each part of the ontology, described in Section 3, we generated a file with the extension *oled*. We performed eight iterations, for the excerpt user feedback elaboration, which are summarized in Table 9. Each row refers to an iteration. For each iteration is reported the version number of the ontology (column *Version*), the number of concepts incorporated and the relations among the concepts (columns *Classes* and *Relations* respectively), the two remaining columns indicate the role of the human expert and of the tools in deriving the ontology version. So for instance if only the *Tool* column is checked in a given row, it means that that version of the ontology was obtained accepting all the changes suggested by the tool, while if both the *Human validator* and *Tool* columns are checked it means that only the changes accepted by the human validator, among those proposed by the tool were implemented. The last version of the user feedback (version 0.8), corresponds to the one depicted in Fig. 7, which is composed of 24 classes and 23 relations.

In the following we discuss in more detail how we evolved our ontology along the defined tool-supported evaluation framework.

- Syntactic quality metric

- * Improving the quality. Table 9 shows the evolution (in terms of number of concepts and relations) that each version of the ontology presented until refining the ontology at version 0.8. The need for evolving the ontology was due to two factors, either because the human validators (ontology engineer and domain expert) agreed on changes or because the validation through the tool was an indicator for evolving. The first row of the table presents the header for each column, i.e., Version,

Table 9

Versions of the excerpt *user feedback elaboration* evolved due to human and/or tool validator

Version	Classes	Relations	Human validator	Tool
V 0.1	10	8	✓	
V 0.2	12	11	✓	
V 0.3	18	15	✓	
V 0.4	21	20	✓	✓
V 0.5	24	23		✓
V 0.6	23	22	✓	✓
V 0.7	22	21	✓	
V 0.8	24	23		

Table 10

Name and frequency of the anti-patterns we found on each version of the ontology of the user feedback elaboration

V 0.1		V 0.2		V 0.3		V 4.0 and V 0.5		V 0.6		V 0.7 and V 0.8	
f	anti-pattern	f	anti-pattern	f	anti-pattern	f	anti-pattern	f	anti-pattern	f	anti-pattern
1	RepRel	1	RepRel	1	HomoFunc		None found		ImpAbs		None found
1	RelRig	1	UndefFormal	1	MultiDep						
				1	RelRig						
				2	RepRel						

Classes, Relations, Human validator and Tool. For instance, the version V 0.3 has 18 classes and 15 relations, the evolution was motivated by the human validators because some necessary concepts were missing. As we can see, we increased gradually the number of concepts on each version until decreasing at the final version based on the criteria of both ontology engineer and domain expert being satisfied with the amount of concepts and relations that represents better the intended world. But also the reduction of concepts and relations was due to the validation done with the tool. Details of the validation are presented in the following.

- * Anti-patterns. We validated each version of our ontology in order to eliminate anti-patterns. Table 10 shows the resulting anti-patterns of each version, the first two rows refers to the headers. The first row presents the version of the ontology and the second row presents the frequency, denoted as f, and the name of the anti-pattern. We see that some anti-patterns appear from version 0.1 to 0.3 and then in 0.6. The consolidation of the ontology started from version 0.4. In the versions 0.1 and 0.2 we only found 2 anti-patterns, while in version 0.3 we found 5 anti-patterns, such as HomoFunc, MultiDep and RelRig with frequency of 1, and RepRel with frequency 2. There are no anti-patterns in version 0.4 and 0.5 but there is one anti-pattern in version 0.6, called ImpAbs.

Below, we briefly explain the anti-patterns found in the versions 0.1, 0.2, 0.3 and 0.6 by using the OLED tool (detailed description of the anti-patterns can be found in the thesis of Sales (2014)):

- * RepRel: the discovery of the Repeatable Relator Instances anti-pattern aims at supporting the modeler in specifying the number of different relators instances that can mediate the exact same set of individuals.
- * UndefFormal: the Undefined Formal Association happens when a formal association is defined between types that do not own or inherit quality properties, i.e., attributes or associations whose types are data types.
- * HomoFunc: the Homogeneous Functional Complex anti-pattern refers to a functional complex type connected to a single part through a «componentOf» relation.
- * MultiDep: the Multiple Relational Dependency anti-pattern aims at investigating a particular set of externally dependent types: the ones that require more than one dependency simultaneously.
- * RelRig: the Relator Mediating Rigid Types anti-pattern occurs whenever a model contains a relator connected to at least one rigid type (stereotyped as «kind», «quantity», «collective», «subkind» or «category») through associations stereotyped as «mediation».
- * ImpAbs: this anti-pattern is present when there is an association R between two classes if at least one of the following options holds: (i) R's source end upper bound multiplicity is equal or greater than 2 and the Class connected to it has 2 or more subtypes; (ii) R's target end

Table 11

Versions of the ontology accepted by the Alloy analyzer to generate instances							
V 0.1	V 0.2	V 0.3	V 0.4	V 0.5	V 0.6	V 0.7	V 0.8
×	✓	✓	×	×	✓	✓	✓

upper bound multiplicity is equal or greater than 2 and the Class connected to it has 2 or more subtypes.

The correction of the anti-patterns involved the modification or deletion of relations types, properties or multiplicity. For example, for solving the anti-pattern *ImpAbs* the OLE tool modified the multiplicity between the classes *Feedback description* and *Attachment* and also the meta-properties of the relation. With the elimination of the anti-patterns we can say that the quality of modeling of our ontology is improved but also that the ontology avoids the unintended instances of states of affairs.

- Essence metric

- * Simulation of instances with Alloy. We present in Table 11 a report of the versions of the ontology that allow a generation of instances. For example, if a version X of the ontology is acceptable to generate instances of concepts with the Alloy analyzer we mark the cell (this means the model is consistent). The first row reports the header of the number of the version, while the second row reports the acceptance (i.e., ✓) or rejection (i.e., ×) of the models to perform the simulation. We observe that the models of versions 0.2, 0.3, 0.6 and 0.7 can generate instances, version 0.7 is stable as well as 0.8 but only version 0.8 satisfies the human validators.

The fact that a version of the ontology can generate instances does not imply that such a version has good quality, because, for example, the version 0.6 contains anti-patterns, meaning that the generated instances can represent unintended states of affairs. The purpose of presenting which version of the ontology are allowed to generate instances with Alloy is to emphasize that the syntactic quality metric and essence metric are complement for ensuring that we are producing a high quality ontology. The Fig. 10 depicts one example of the instances that Alloy is able to generate. The rectangles represent the objects that exist in the real world and the ovals are the properties associated to the objects.

In this example we can see that there are two different instances of feedback provider who use the same software application. There exists a feedback collector that performs the feedback collection of two different topics. The shared descriptions of user feedback are available through the feedback channel open forum, and these descriptions are collected by the feedback collector. In this figure each description is presented as unstructured text and each one contains as attachment a video. We can see that the instances of the figure shows a possible representation of the real world, which is logically correct and complies with the ontology we have described.

5.2. Discussion

From the results of the evaluation we can see that the evolution of the ontology of the user feedback elaboration has been positive in terms of syntactic quality of the modeling. However, we need to make a compromise between the amount of essential concepts and relations, and the number of anti-patterns that we add with each elimination or addition of concepts from version 0.1 to 0.6, as it can be observed in Table 10.

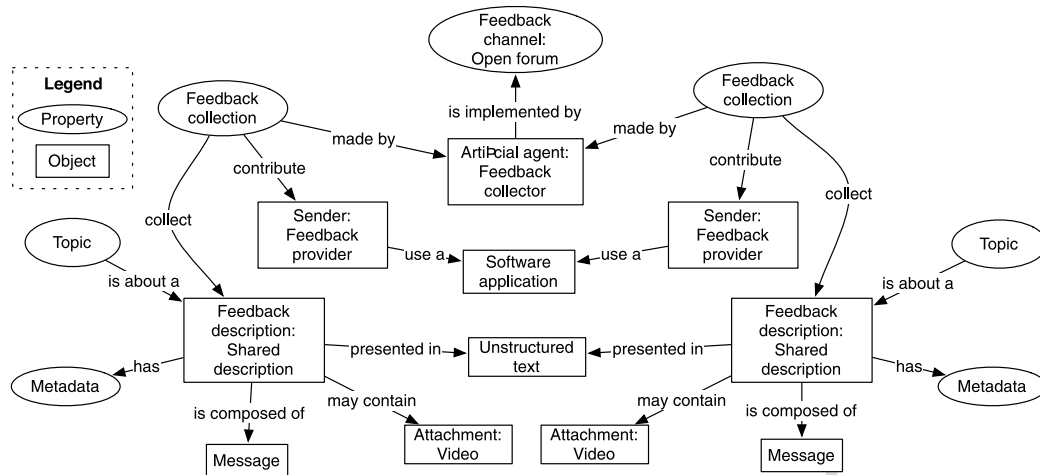


Fig. 10. Example of a world with instances generated with the Alloy analyzer corresponding to the excerpt user feedback elaboration.

The version 0.7 already does not have anti-patterns and this can be interpreted as a stable version that can be evolved by adding concepts that may extend the scope of the ontology if needed. Despite that the version 0.7 is stable, the domain expert and the ontology engineer (first and third authors, respectively) have coincided with another iteration with the purpose of keeping aligned the progress achieved so far in the pragmatic research (automatization of the speech act annotation) thus including this knowledge, acquired by the domain expert, into the ontology. The decision of including new concepts and relations have been consciously discussed by the domain expert and ontology engineer in order to not alter dramatically the quality of the models but describing precisely the concepts involved during the elaboration and analysis of online user feedback.

Roughly, the number of iterations of the ontology correspond to the number of sessions for discussing the concepts, in total we have engaged in about 19 sessions (the three authors), each one with a duration of minimum 1 h maximum 2 h. Approximately, 8 sessions were used to discuss the concepts regarding the user feedback elaboration, 7 sessions to discuss about the concepts of the feedback analysis, and 4 sessions for discussing the extension to the communication ontology. It took less sessions to define the extension to the communication ontology since the third author of this paper is the domain expert, of the communication ontology, and is the ontology engineer.

In addition, something important to notice from Tables 10 and 11 is that even though the versions 0.4 and 0.5 do not present any anti-pattern, it does not mean a validation of the instances in the real world, as the Alloy analyzer performs. This finding indicates that an evolution of the ontology must be carefully thought in terms of maintaining a logical relationship between concepts when concepts are included or deleted.

Concerning the essence metric the simulation of the instances shows correct representations of the world, but some constraints are needed. For example, we have identified in the application scenarios (see Section 4) that some feedback channels restrict the presentation format of the feedback descriptions. For this reason, we would probably add axioms to the ontology for representing such a constraint between the presentation format and feedback channel. Another possible axiom to be added regards the message that carries the propositional content of a communicative act, and the specific number of speech acts

contained in a message. We have identified in the scenarios that one message (linguistically represented as a sentence) can contain from 1 to 2 speech acts. This leads to the need of an axiom to specify it.

6. Related work

Our ontology represents a novel conceptual framework that describes all the concepts found in the domain of user feedback. Besides this, this framework can be exploited for supporting the analysis of user feedback, when considering it as a textual artifact. Up to our knowledge there are no research works dealing with a conceptualization of user feedback, but there is an emerging research aiming at developing taxonomies either of the user feedback dimensions, developed by Groen et al. (2015), or related to the feedback classification (called taxonomy of issue types) described by Merten et al. (2015). Therefore, the considered related work presented in this section concerns the use of ontologies for enabling the communication among distributed teams, supporting a phase of the software development or ontologies for describing the software as an artifact or ontologies used in requirements engineering.

Our ontology extends the communication ontology, which is part of the ontology CONTO, that aims at giving foundation to the concepts of the collaboration, communication and coordination processes in software development (Oliveira et al., 2007). In line with this, the work by Breslin et al. (2006) presents the ontology SIOC –Semantically Interlinked Online Community– that tackles the problem of enabling the location of relevant information dispersed in different platforms used by online communities, by identifying and modeling the concepts found in online discussion methods. For instance, some concepts are: *people, groups, role, posts, forums, locations, categories*, etc. The SIOC ontology identifies the main community concepts and enables the linking of these concepts by modeling the relationships between them. Our ontology differs from SIOC on presenting a depth characterization of user feedback as an artifact containing valuable information. However, we see the potential of connecting our ontology to SIOC by means of concepts such as *post* or *role*, thus expanding the scope of our ontology.

A work describing the application of ontologies in collaborative software development is presented by Happel et al. (2010). They describe the benefits of ontologies (both as conceptual and technical artifacts) to improve the communication and understanding between distributed teams, due to ontologies can enable the semantic annotation in artifacts. In this paper the authors describe some standard collaboration ontologies such as the FOAF –Friend-of-a-Friend–. This ontology provides mainly a vocabulary regarding the social networks in a decentralized manner, the ontology is specified with RDF and defines basic concepts such as *person, group* or *organization*, and properties such as *name, knows, workplace-Homepage*. Another ontology is DOAP –Description of a Project– whose core concept is the *software project* with some properties such as *category, license* and *bug tracking URL*.

Concerning the ontologies for/of software we can mention the one proposed by Oberle et al. (2009) and the ontology of Wang et al. (2014). On one side, Oberle et al. (2009) present the core software ontology as a reference for specifying the intended meanings of software terms as precisely as possible. This ontology is founded on the foundational ontology DOLCE and provides a formalization of concepts such as *ComputationalObject, ComputationalActivity, Data, AbstractData*, among other concepts. In this work, the authors perform a deep ontological analysis of what is a software and the elements involving it (e.g., interface, method, inputs, outputs). Contrary, the user feedback ontology accounts for explaining the origin of its elaboration from an ontological and linguistic perspective.

On the other side, Wang et al. (2014) aim at establishing a well-founded ontological framework to understand the different cases of software change rationale across the whole software engineering life-cycle. As a first step the authors discuss on the need to understand what software is, specifically from

the requirements engineering perspective. In addition, they discuss concepts regarding the versioning of software that might give foundation for software evolution. With our ontology we first try to understand the source, i.e. user feedback, of such changes that trigger the software evolution.

Another ontology that tackles the problem of software changes is the ontology of software maintenance (Kitchenham et al., 1999). The authors urged for the need of understanding the maintenance domain, thus providing a conceptual framework talking about the domain and the factors that affect maintenance. An important benefit of having such a maintenance ontology is to provide a standard framework to assist the reporting of empirical studies in such a way that they can be classified, understood and replicated. From this ontology we have noticed that the authors employ the concepts *Enhancement* and *Correction* as specializations of *Modification Activity*. Moreover, *Enhancement* is further refined into *New Requirements*, *Changed Requirements* and *Changed Implementation*. While in our ontology of user feedback we consider *Enhancement* as a synonym of *New feature request*, and *Correction* as a synonym of *Bug report* for defining types of *Feedback classification*.

With respect to the software development process, we can mention the software process ontology (SPO) described by de Oliveira Bringuente et al. (2011) and reengineered by using the foundational ontology UFO. The SPO aims at establishing a common conceptualization for software organizations to talk about software process and it is divided into four sub-ontologies (i.e., activity, resource, procedure, and software process). And the sub-ontology software process reengineered along the UFO to address some issues and to improve the definition and distinction of concepts such as, for example, *Software Process* and *Project Process* or *Activity Scheduled* and *Activity Occurrence*. The main goal is to support software project planning in an standardized way.

Concretely, the use of ontologies during the phase of requirements engineering has been investigated through a systematic literature review by Dermeval et al. (2015), concluding, for example, that ontologies support the domain knowledge representation to guide requirements elicitation, alleviate the requirements communication problem as well as assist in requirements management/evolution.

As an example of research work, the core ontology for requirements by Jureta et al. (2009) aims at reformulating the requirements problem stated by Zave and Jackson (1997) by revising the concepts of requirements, domain assumption, and specification from an ontological point of view. The authors claim that, among other issues, the current characterization of the requirements problem is generic in guiding the construction of requirements formalisms for the representation of information relevant during the requirements elicitation phase. This work has is grounded in the communication between the stakeholders and the software engineer, using as baseline the speech act theory. Jureta et al. only characterize the communication of the main categories of speech acts, namely, assertive, directive, commissive, expressive, declarative and representative declarative. While we emphasize the importance of specializing the speech acts into more concrete classes that are better representative of written natural language.

An approach that aims at supporting developers of information systems by providing suitable ontologies to guide the software process, specifically for verifying a requirements model is the approach presented by Beydoun et al. (2014). This approach supports the verification of the consistency of a requirements model with the use of an ontology. The verification is performed by creating an ontology from this model, find the appropriate ontologies, related to the requirements model, to be adapted and integrated. The ultimate objective is to use the found ontologies to follow an ontology-based information system development. Omoronyia et al. (2010) present an approach to build a domain ontology from existing technical standards, with the purpose of guiding the requirements elicitation process. In this work the authors present three semantic features for enriching axioms of the ontology-to-be in order to

build a proper ontology suitable for requirements elicitation. The overall approach includes the performance of text analysis, which is rule-based and applies NLP techniques. The approach involves steps such as manual textual document pre-processing; a semi-automated process that supports the completion of words to avoid ambiguity; concept clustering to perform a lexical similarity matching approach to discard repetitive concepts and then generate a taxonomy tree.

7. Conclusion and future work

In this paper we presented a novel user feedback ontology specified in OntoUML (Guizzardi, 2005). We focused on online feedback given by users upon their experience in using a software service or application. Differently from customer's evaluation that is expressed in terms of quality-price comparison of similar products, recommendation to people to buy (or not) the product or as a summary of the product's features. User feedback usually concerns questions on technical details about the software, experience of use, reporting flaws or the request of new features.

We characterized user feedback in terms of speech acts to emphasize that performative-verb phrases contained in user feedback let interpret the original intentions of the user, e.g. to request a new feature for the software application she is using. Though most of the concepts of the proposed ontology apply to online user feedback in general, we added specific entities that are needed when we consider user feedback in software application domain.

The proposed ontology has been used to give a full account of the concepts needed to describe user feedback and its analysis in several application scenarios. These scenarios consider different presentation formats of user feedback, namely, unstructured, semi-structured and structured text, as well as manual and semi-automated analysis of user feedback, as performed in requirements engineering tasks. We described how we evolved the ontology through an iterative validation process with the goal of improving its modeling quality, by eliminating possible anti-patterns in OntoUML models. Moreover, we used the Alloy analyzer for generating instances of the concepts of the user feedback ontology and validate a true correspondence of the concepts and their relations against the real world.

The following considerations aim at clarifying the scope of the work presented and the applicability of the proposed ontology. Worth to point out is that while in the human computer interaction literature (see for instance Bastide et al. (1998) and Arteaga (2002)) the concept of feedback is used to describe what happens during the interaction of a user with a software application, in our work we are dealing with a different phenomenon. Namely, we address the user feedback in the domain of software development wherein users (sender) provide feedback descriptions (textual) to the software analyst (receiver), upon having interacted with the software application.

With respect to the feedback cycle in software evolution, which we mentioned as background of our work in Section 2, we shall recall that in the literature, e.g. Brun et al. (2009), it is commonly defined as a four-phase cycle including the collection, analysis, planning and act phases. Our ontology aims first at supporting the understanding of the collection and the analysis phases of the feedback cycle, by means of representing how the online user feedback is elaborated and analyzed in terms of speech acts. The last two phases of the cycle, when the analyst plans and takes the appropriate actions in order to reach the user's desired performance of the software application, are not discussed in this paper.

Focusing on the applicability of the proposed ontology from the perspective of the language used, we are currently considering English due to two facts: first, we rely on the speech act theory which has been proposed for the English language; second, because the type of communication among people working

in software development environments is performed mainly in English. However, we believe that our approach might be customizable to Latin-based languages.

As future work we want to refine the parts of the ontology corresponding to the speech acts taxonomy and the user feedback analysis, respectively, with the purpose to code the ontology (for instance in OWL) at support of semantic annotation of speech acts. This result will contribute to extend the method and the analysis model presented in a previous work (Morales-Ramirez et al., 2014a). As further application purpose of the proposed ontology we will explore the possibility of using it as a framework to support a meta review of the new literature on user feedback and classify it according to the goal of gathering or analyzing user feedback.

Acknowledgements

We thank Giancarlo Guizzardi for his time to discuss the initial stage of this research work and for his suggestions to improve it. We also want to thank Tiago Prince Sales, Bernardo Ferreira Bastos Braga, John Guerson and Maria das Graças da Silva Teixeira for their technical support regarding the OLED tool. This work is a result of the SUPERSEDE project, funded by the H2020 EU Framework Programme under the agreement number 644018 and also partially supported by CAPES/CNPq (Grant number 402991/2012-5) and CNPq (Grant numbers 461777/2014-2 and 485368/2013-7).

References

- Arteaga, J.M. (2002). A visual feedback taxonomy for interactive systems. In C. Pribeanu and J. Vanderdonck (Eds.), *Task Models and Diagrams for User Interface Design: Proceedings of the First International Workshop on Task Models and Diagrams for User Interface Design – TAMODIA 2002*, Bucharest, Romania, 18–19 July 2002 (pp. 49–54). Bucharest: INFOREC Publishing House.
- Austin, J. (1962). *How to Do Things with Words*. Oxford.
- Bach, K. & Harnish, R.M. (1979). *Linguistic Communication and Speech Acts*. Cambridge, MA: The MIT Press.
- Bastide, R., Palanque, P., Le, D.-H. & Muñoz, J. (1998). Integrating rendering specifications into a formalism for the design of interactive systems. In P. Markopoulos and P. Johnson (Eds.), *Design, Specification and Verification of Interactive Systems'98, Eurographics* (pp. 171–190). Vienna: Springer.
- Beydoun, G., Low, G., García-Sánchez, F., Valencia-García, R. & Martínez-Béjar, R. (2014). Identification of ontologies to support information systems development. *Information Systems*, 46, 45–60.
- Blakemore, D. (2003). Discourse and relevance theory. In *The Handbook of Discourse Analysis*.
- Bosch-Sijtsema, P. & Bosch, J. (2014). User involvement throughout the innovation process in high-tech industries. *Journal of Product Innovation Management*, 32(5), 793–807.
- Breslin, J.G., Decker, S., Harth, A. & Bojars, U. (2006). SIOC: An approach to connect web-based communities. *International Journal of Web Based Communities*, 2(2), 133–142.
- Brun, Y., Di Marzo Serugendo, G., Gacek, C., Giese, H., Kienle, H., Litoiu, M., Müller, H., Pezzè, M. & Shaw, M. (2009). Engineering self-adaptive systems through feedback loops. In *Software Engineering for Self-Adaptive Systems* (pp. 48–70). Berlin, Heidelberg: Springer.
- Burton-Jones, A., Storey, V.C., Sugumaran, V. & Ahluwalia, P. (2005). A semiotic metrics suite for assessing the quality of ontologies. *Data & Knowledge Engineering*, 55(1), 84–102. (Special Issue: Natural Language and Database and Information Systems NLDB 03.)
- Carreño, L.V.G. & Winbladh, K. (2013). Analysis of user comments: An approach for software requirements evolution. In D. Notkin, B.H.C. Cheng and K. Pohl (Eds.), *ICSE* (pp. 582–591). IEEE/ACM.
- Cleland-Huang, J., Dumitru, H., Duan, C. & Castro-Herrera, C. (2009). Automated support for managing feature requests in open forums. *Communications of the ACM*, 52(10), 68–74.
- de Oliveira Bringunte, A.C., de Almeida Falbo, R. & Guizzardi, G. (2011). Using a foundational ontology for reengineering a software process ontology. *Journal of Information and Data Management*, 2(3), 511–526.
- Debusse, J.C.W. & Lawley, M. (2015). Benefits and drawbacks of computer-based assessment and feedback systems: Student and educator perspectives. *British Journal of Educational Technology*. doi:10.1111/bjet.12232.

- Dermeval, D., Vilela, J., Bittencourt, I., Castro, J., Isotani, S., Brito, P. & Silva, A. (2015). Applications of ontologies in requirements engineering: A systematic review of the literature. *Requirements Engineering*, 14 February 2015. doi:10.1007/s00766-015-0222-6.
- Duranti, A. (2006). The social ontology of intentions. *Discourse Studies*, 8(1), 31–40.
- Esuli, A. & Sebastiani, F. (2006). SENTIWORDNET: A publicly available lexical resource for opinion mining. In *Proceedings of the 5th Conference on Language Resources and Evaluation (LREC'06)* (pp. 417–422).
- Fernandes, P.C.B., Guizzardi, R.S.S. & Guizzardi, G. (2011). Using goal modeling to capture competency questions in ontology-based systems. *Journal of Information and Data Management*, 2(3), 527–540.
- Godfrey, M. & German, D. (2008). The past, present, and future of software evolution. In *Proceedings of the 2008 Frontiers of Software Maintenance, FoSM 2008* (pp. 129–138).
- Groen, E., Doerr, J. & Adam, S. (2015). Towards crowd-based requirements engineering a research preview. In S.A. Fricker and K. Schneider (Eds.), *Requirements Engineering: Foundation for Software Quality*. LNCS (Vol. 9013, pp. 247–253). Springer.
- Guarino, N. & Welty, C. (2009). An overview of OntoClean. In S. Staab and R. Studer (Eds.), *Handbook on Ontologies*. International Handbooks on Information Systems (pp. 201–220). Berlin, Heidelberg: Springer.
- Guizzardi, G. (2005). Ontological foundations for structural conceptual models. PhD thesis, University of Twente, The Netherlands.
- Guizzardi, G. (2014). Ontological patterns, anti-patterns and pattern languages for next-generation conceptual modeling. In *Conceptual Modeling – Proceedings of the 33rd International Conference, ER 2014*. LNCS, Atlanta, GA, USA, 27–29 October 2014 (Vol. 8824, pp. 13–27). Springer.
- Guizzardi, G., Baião, F.A., Lopes, M. & de Almeida Falbo, R. (2010). The role of foundational ontologies for domain ontology engineering: An industrial case study in the domain of oil and gas exploration and production. *International Journal of Information System Modeling and Design*, 1(2), 1–22.
- Guizzardi, G., de Almeida Falbo, R. & Guizzardi, R.S.S. (2008). Grounding software domain ontologies in the unified foundational ontology (UFO): The case of the ODE software process ontology. In M. Lencastre, J.F. e Cunha and A. Valecillo (Eds.), *CIBSE* (pp. 127–140).
- Guizzardi, R.S.S., Morales-Ramirez, I. & Perini, A. (2014). A goal-oriented analysis to guide the development of a user feedback ontology. In *Proceedings of the 7th International i* Workshop Co-located with CAiSE*. CEUR Workshop Proceedings, Thessaloniki, Greece, 16–17 June 2014 (Vol. 1157). Available at: CEUR-WS.org.
- Guzman, E. & Maalej, W. (2014). How do users like this feature? A fine grained sentiment analysis of App reviews. In T. Gorschek and R.R. Lutz (Eds.), *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, Karlskrona, Sweden, 25–29 August 2014 (pp. 153–162). IEEE.
- Happel, H., Maalej, W. & Seedorf, S. (2010). Applications of ontologies in collaborative software development. In I. Mistrík, A. van der Hoek, J. Grundy and J. Whitehead (Eds.), *Collaborative Software Engineering* (pp. 109–129). Springer.
- Hattie, J. & Timperley, H. (2007). The power of feedback. *Review of Educational Research*, 77(1), 81–112.
- Jackson, D. (2006). *Software Abstractions: Logic, Language, and Analysis*. The MIT Press.
- Jawaheer, G., Weller, P. & Kostkova, P. (2014). Modeling user preferences in recommender systems: A classification framework for explicit and implicit user feedback. *ACM Transactions on Interactive Intelligent Systems*, 4(2), 8.
- Jureta, I., Mylopoulos, J. & Faulkner, S. (2009). A core ontology for requirements. *Applied Ontology*, 4(3,4), 169–244.
- Kitchenham, B.A., Travassos, G.H., von Mayrhauser, A., Niessink, F., Schneidewind, N.F., Singer, J., Takada, S., Vehvilainen, R. & Yang, H. (1999). Towards an ontology of software maintenance. *Journal of Software Maintenance: Research and Practice*, 11(6), 365–389.
- Lehman, M. (1980). Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE*, 68(9), 1060–1076.
- Maalej, W., Happel, H.-J. & Rashid, A. (2009). When users become collaborators: Towards continuous and context-aware user input. In *Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications, OOPSLA'09* (pp. 981–990). New York, NY, USA: ACM.
- Madhavji, N.H., Fernández-Ramil, J.C. & Perry, D.E. (Eds.) (2006). *Software Evolution and Feedback: Theory and Practice*. John Wiley & Sons Ltd.
- Masli, M. & Terveen, L.G. (2014). Leveraging the contributory potential of user feedback. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing, CSCW'14* (pp. 956–966). New York, NY, USA: ACM.
- Merten, T., Mager, B., Hübner, P., Quirchmayr, T., Paech, B. & Bürsner, S. (2015). Requirements communication in issue tracking systems in four open-source projects. In *Joint Proceedings of REFSQ-2015 Workshops*. CEUR Workshop Proceedings, Essen, Germany, 23 March 2015 (Vol. 1342, pp. 114–125). Available at: CEUR-WS.org.
- Moghaddam, S. (2015). Beyond sentiment analysis: Mining defects and improvements from customer feedback. In *Advances in Information Retrieval – Proceedings of the 37th European Conference on IR Research, ECIR 2015*. LNCS, Vienna, Austria, 29 March–2 April 2015 (Vol. 9022, pp. 400–410).

- Morales-Ramirez, I. (2015). Exploiting online user feedback in requirements engineering. PhD thesis, University of Trento, Italy.
- Morales-Ramirez, I., Perini, A. & Ceccato, M. (2014a). Towards supporting the analysis of online discussions in OSS communities: A speech-act based approach. In *Information Systems Engineering in Complex Environments – CAiSE Forum 2014, Selected Extended Papers*. LNBP, Thessaloniki, Greece, 16–20 June 2014 (Vol. 204, pp. 215–232). Springer.
- Morales-Ramirez, I., Perini, A. & Guizzardi, R.S.S. (2014b). Providing foundation for user feedback concepts by extending a communication ontology. In *Conceptual Modeling – Proceedings of the 33rd International Conference, ER 2014*. LNCS, Atlanta, GA, USA, 27–29 October 2014 (Vol. 8824, pp. 305–312). Springer.
- Mory, E.H. (2004). Feedback research revisited. *Handbook of Research on Educational Communications and Technology*, 45(1), 745–784.
- Noy, N.F. & McGuinness, D.L. (2001). Ontology development 101: A guide to creating your first ontology. Technical report.
- Oberle, D., Grimm, S. & Staab, S. (2009). An ontology for software. In S. Staab and R. Studer (Eds.), *Handbook on Ontologies*. International Handbooks on Information Systems (pp. 383–402). Berlin, Heidelberg: Springer.
- Oliveira, F.F., Antunes, J.C. & Guizzardi, R.S. (2007). Towards a collaboration ontology. In *Proceedings of the 2nd Brazilian Workshop on Ontologies and Metamodels for Software and Data Engineering*.
- Omoronyia, I., Sindre, G., Stålhane, T., Biffi, S., Moser, T. & Sunindyo, W. (2010). A domain ontology building process for guiding requirements elicitation. In R. Wieringa and A. Persson (Eds.), *Proceedings of the 16th International Working Conference, REFSQ 2010*. LNCS (Vol. 6182, pp. 188–202). Berlin, Heidelberg: Springer.
- Ordenes, F.V., Theodoulidis, B., Burton, J., Gruber, T. & Zaki, M. (2014). Analyzing customer experience feedback using text mining: A linguistics-based approach. *Journal of Service Research*. doi:10.1177/1094670514524625.
- Pagano, D. & Maalej, W. (2013). User feedback in the AppStore: An empirical study. In *2013 IEEE 21st International Requirements Engineering Conference (RE)*, Rio de Janeiro, Brazil, 15–19 July 2013 (pp. 125–134). IEEE Computer Society.
- Proynova, R. & Paech, B. (2013). Factors influencing user feedback on predicted satisfaction with software systems. In *Requirements Engineering: Foundation for Software Quality – Proceedings of the 19th International Working Conference, REFSQ 2013*. LNCS, Essen, Germany, 8–11 April 2013 (Vol. 7830, pp. 96–111). Springer.
- Reforgiato Recupero, D., Presutti, V., Consoli, S., Gangemi, A. & Nuzzolese, A. (2015). Sentilo: Frame-based sentiment analysis. *Cognitive Computation*, 7(2), 211–225.
- Sabou, M. & Fernandez, M. (2012). Ontology (network) evaluation. In M.C. Suárez-Figueroa, A. Gómez-Pérez, E. Motta and A. Gangemi (Eds.), *Ontology Engineering in a Networked World* (pp. 193–212). Berlin, Heidelberg: Springer.
- Sales, T.P. (2014). Ontology validation for managers. Master's thesis, Universidade Federal do Espírito Santo.
- Schneider, K. (2011). Focusing spontaneous feedback to support system evolution. In *2011 IEEE 19th International Requirements Engineering Conference (RE)*, Trento, Italy, 29 August–2 September 2011 (pp. 165–174). IEEE Computer Society.
- Searle, J.R. (1969). *Speech Acts: An Essay in the Philosophy of Language* (Vol. 626). Cambridge Univ. Press.
- Searle, J.R. (1983). *Intentionality: An Essay in the Philosophy of Mind* (Number 143). Cambridge Univ. Press.
- Seyff, N., Graf, F. & Maiden, N.A.M. (2010). Using mobile RE tools to give end-users their own voice. In *2010 IEEE 18th International Requirements Engineering Conference (RE)*, Sydney, New South Wales, Australia, 27 September–1 October 2010 (pp. 37–46). IEEE Computer Society.
- Seyff, N., Ollmann, G. & Bortenschlager, M. (2014). AppEcho: A user-driven, in situ feedback approach for mobile platforms and applications. In *Proceedings of the 1st International Conference MOBILESoft 2014*, Hyderabad, India, 2–3 June 2014 (pp. 99–108). ACM.
- Wang, X., Guarino, N., Guizzardi, G. & Mylopoulos, J. (2014). Towards an ontology of software: A requirements engineering perspective. In *8th International Conference, FOIS 2014*, Frontiers in Artificial Intelligence and Applications, Rio de Janeiro, Brazil, 22–25 September 2014 (Vol. 267, pp. 317–329). IOS Press.
- Wilson, D. & Sperber, D. (2002). *Relevance Theory* (pp. 607–632). Blackwell.
- Zave, P. & Jackson, M. (1997). Four dark corners of requirements engineering. *ACM Transactions on Software Engineering and Methodology*, 6(1), 1–30.