Assisted modelling over social networks with Socio

Sara Pérez-Soler, Esther Guerra, Juan de Lara
Modelling & Software Engineering Research Group
http://miso.es
Computer Science Department
Universidad Autónoma de Madrid (Spain)
e-mail: {sara.perezs, esther.guerra, juan.delara}@uam.es

***Abstract*—Social networks are intensively used nowadays for both leisure and work. They have become a natural communication mechanism which helps users in coordinating and collaborating in their daily life activities.**

To profit from their pervasive use, we propose Socio: a modelling assistant that seamlessly integrates across several social networks, like Telegram or Twitter. Socio is a modelling bot that can interpret natural language sentences and create meta-models out of them. It provides traceability of design decisions and statistics of user contributions. A video showcasing the tool is available at https://saraperezsoler.github.io/ModellingBot/.

***Index Terms*—Meta-modelling, modelling bots, social networks, natural language processing**

I. Introduction

Social networks are becoming an important part of our daily digital lives, where we use them to keep in touch with friends and organize leisure activities. They are also gaining relevance as a mechanism to disseminate information and to cooperate and coordinate in work tasks. In particular, social media is increasingly used in software engineering, e.g., to support the formation of ecosystems around particular concepts and technologies; to participate in online development communities; and to disseminate technologies and crowdsource content [1], [2].

Given the widespread use of social networks, our goal is to exploit them for collaborative modelling. Hence, we introduce the concept of *modelling bot*, able to interpret natural language (NL), assist users in creating models, and which integrates with minimum disruption into the natural communication mechanism that micro-blogging based social networks – like Twitter or Telegram – offer. Our approach is lightweight, as it can be used in mobility and it does not require installing any dedicated modelling tool, but users can employ the social network client they are familiar with. It can be used by teams of engineers, likely distributed, to create models and enhance their global coordination. It fosters collaboration with domain experts, who might be unfamiliar with modelling tools, by interpreting domain requirements expressed in NL. Finally, it has the potential to facilitate the crowdsourcing of design decisions. While bots are starting to be used to help in software engineering activities like documentation or user support [3], to our knowledge, ours is the first proposal for a modelling assistant bot.

In this tool demo paper, we describe our modelling bot Socio [4] which works across several social networks like Twitter and Telegram. The tool interprets messages from the social network using the Stanford NL parser [5], the WordNet [6] lexical database to identify synonyms, and an extensible set of domain requirements extraction rules [7] that are responsible for growing the model to reflect the messages information. The bot keeps track of all model changes and their provenance, and provides statistics of the participation of each user and their percentage of model authorship. The constructed models can be validated, and exported to the EMF format. While Socio has been designed to help in both modelling and meta-modelling, the current version of the tool only supports the creation of meta-models.

The remaining of this paper is organized as follows. First, Section II describes our approach to NL processing, and Section III presents the architecture and main features of Socio. Next, Section IV compares with related research. Finally, Section V concludes with prospects for future work.

II. Approach

The goal of our tool is to provide modelling assistance integrated within widely used discussion and collaboration mechanisms such as social networks. For this purpose, our modelling bot integrates seamlessly in the social network as another participant, and the interaction with the bot is based on NL to keep a "sense of flow" as there is no need to switch tools for discussing and modelling. Figure 1 shows a scheme of our approach. At any moment, users can send messages directed to the bot. The way to address the bot varies slightly depending on the particularities of the social network: in Twitter, the message needs to mention the username of the bot (@ModellingBot), while in Telegram, the bot must belong to the modelling group and a command starting by "/" needs to be issued. The bot offers a suite of commands for model management (create a new model, show all existing models, update a model, validate a model, download a model) and to obtain statistics. We will explain these commands in Section III, while in the remainder of this section we will focus on the handling of NL messages expressing domain requirements. Messages not directed to the bot are just regular

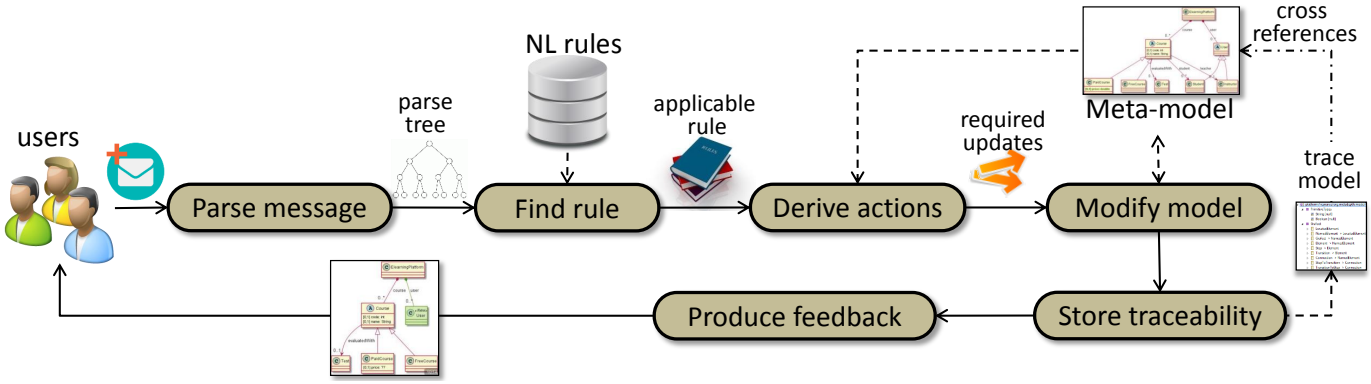


Fig. 1: Processing a NL message for model update

messages used for discussion and coordination within the modelling session.

When the bot receives a message, it uses the Stanford parser to produce a parse tree. Each word in this tree is tagged with its role in the sentence, like noun (singular or plural) or verb (past, gerund, past participle, 3rd person singular present, non-3rd person singular present). We have created a set of extraction rules that detect certain phrase structures of interest in the tree and deduce domain requirements that trigger model updates. The extraction rules are based on the work by Arora and colleagues [7], adapted to our context. The rule set is extensible, and we currently consider the following ones:

- **Verb to be**. These phrases may indicate a subclassing relation (e.g., "users can be students or teachers"), the type of a feature (e.g., "price is double") or the abstractness of a class (e.g., "course is abstract").
- **Verb to have (and synonyms)**. These phrases identify features (attributes or references) of a class. We also consider the Saxon genitive.
- **Transitive verbs**. When a phrase with a verb, subject and direct object is identified, this rule triggers the creation of classes for the subject and direct object (if they do not exist yet), and a reference for the verb.
- **Contain (and synonyms)**. These phrases signal the need for a composition relation between two classes (e.g., "an e-learning platform is made of courses").
- **Add (and synonyms)**. Imperative phrases using "add" or a synonym verb result in the creation of classes or features (e.g., "add name to user").
- **Remove (and synonyms)**. Similar to the previous rule, but for removing elements.

Each rule has a priority, equal to its position in the previous list. In this way, if several rules are applicable, only the one with the highest priority is selected. Then, the selected rule checks the state of the meta-model and produces the necessary meta-model update actions. In this step, rules make use of WordNet to detect synonyms, and take into account grammatical number to allow flexibility when referring to existing classes (e.g., an existing class "Teacher" can be referred later as "Instructors"). The actions are generated explicitly so that their effects can be undone and redone. Supported actions include creating and removing classes, making them abstract or concrete, adding and deleting features, and changing the type of a feature. After processing a message, the updated meta-model may lack some information, like the type of an attribute, or whether a feature is an attribute or a reference. These design decisions remain open and need to be resolved before the model is deemed valid.

The message, the user issuing the message and the performed actions are stored in a traceability model. This allows keeping accurate record of the provenance and rationale of every meta-model element, as well as generating different statistics. Once the meta-model and the trace model have been updated, the bot sends a picture of the meta-model to all users, where the impacted elements are highlighted in a different colour.

1) Example.: Figure 2 illustrates the processing of several NL messages used to build a meta-model for e-learning systems. The first sentence triggers the rule for the verb "to be", and results in the creation of two subclasses of Course. In the second sentence, the user uses a transitive verb ("courses are evaluated with a test"), which yields the creation of a new class Test and a reference evaluatedWith. The bot follows accepted naming conventions: upper camel case for classes and lower camel case for features. Moreover, it also checks the grammatical number of the words to assign an appropriate cardinality to features (e.g., [0..1] in reference evaluatedWith).

III. Tool Support

We have built a modelling bot called Socio to support the presented approach. It works on Telegram and Twitter, though it is designed to be extensible with further social networks and NL rules. The bot stores meta-models in Ecore format, and the traceability data as EMF models. The pictures of meta-models are generated with PlantUML. A video showcasing its use and some examples are available at https://saraperezsoler.github.io/ModellingBot/.

Figure 3 shows some screenshots in the interaction with Socio to build and validate a meta-model for e-learning systems. In the first place (not shown), a Telegram group that includes the participants and the bot needs to be created. In

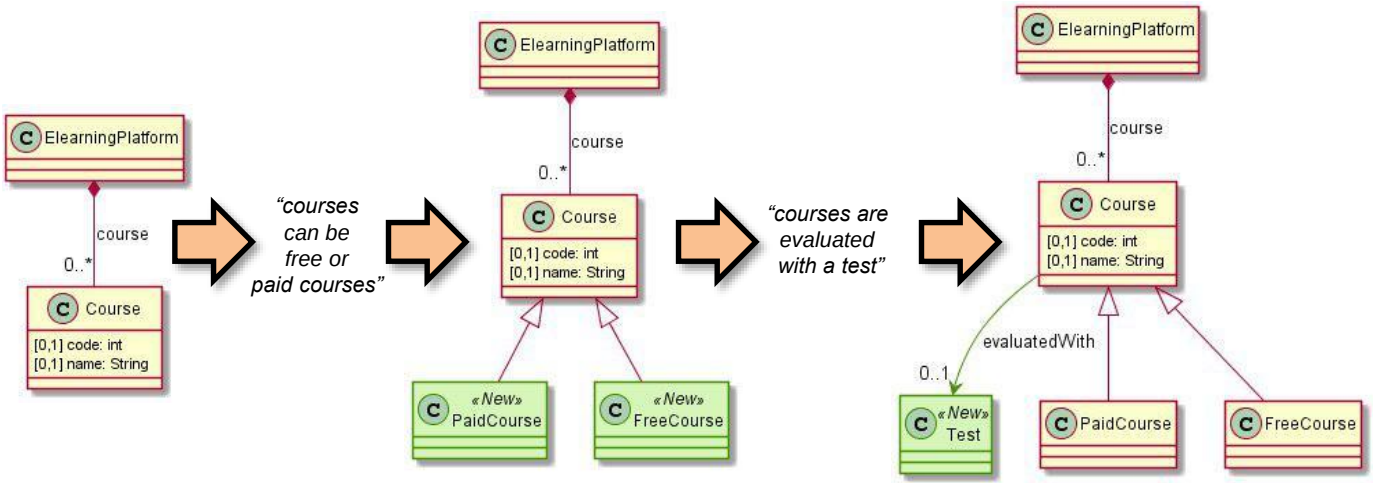


Fig. 2: NL interaction sequence, and corresponding meta-model updates

Figure 3a, the bot shows all available commands, and then, one participant creates a modelling project using the /newproject command.

In Figure 3b, a participant sends a NL message to the bot using the command /talk. The bot interprets the message to deduce domain requirements, updates the current meta-model version accordingly, and returns a picture of the updated meta-model with the created and updated elements highlighted in green. Figure 3c shows a similar interaction using Twitter. In this case, the bot username (@ModellingBot) and the project name (#learningplatform) need to be mentioned. The created attribute (code) is shown in green. After some interactions, one participant validates the model (Figure 3d), and the bot reports there is an error because the type of attribute PaidCourse.price is missing. At any moment, the meta-model can be downloaded in Ecore format using command /get (Figure 4a). The downloaded meta-model can then be used within Eclipse (Figure 4b).

The interaction with Socio is recorded in a traceability model. This can be used to understand the rationale of every decision and analyse user contributions. In particular, Socio offers the following statistics: messages sent by one or all users, meta-model update actions done by one or all users, and percentage of meta-model authorship. They are available through the /history command (see Figure 5a). As an example, Figure 5b shows the number of messages directed to the bot from all users along time, while Figure 5c shows the percentage of authorship. In addition, it is also possible to obtain a more detailed history of the messages sent by each user and their consequences.

IV. Related work

There are several approaches for collaborative modelling or meta-modelling [8]. Some recent examples include Collaboro [9] and SPACEclipse [10]. However, these works do not consider modelling assistants or NL as an interaction mechanism, and they do not rely on social networks but on platforms like Eclipse, which requires technical expertise. Instead, our approach integrates a modelling bot within a social network, so that users do not need to switch between discussion, coordination and modelling tools.

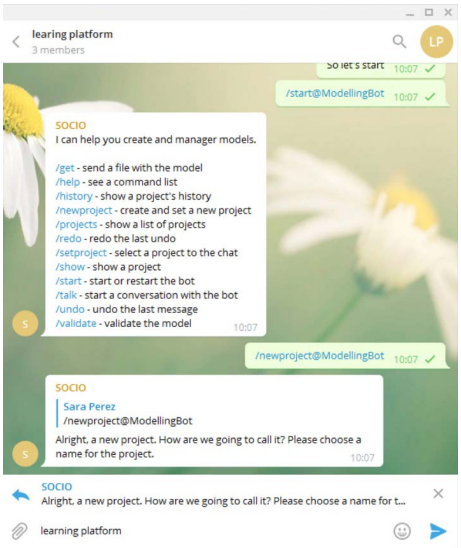
Social networks have been recognised to have a high-impact research potential in software engineering [1], [11]. In fact, we are witnessing an increasing use of bots to automate certain software engineering tasks, like DevOps activities or user support bots [3]. The general goal of their use is to improve efficiency (do things faster) and effectiveness (complete tasks towards meaningful goals). Socio targets improving effectiveness of meta-model creation, lowering the barrier of participation to non-technical people.

Several researchers have proposed different models of crowdsourcing for software design activities. For example, in [12], the authors use microtask crowdsourcing for parallelising the construction of morphological charts, a design technique to represent decision points and alternative solutions. Their experiments show the feasibility of using crowdsourcing to generate a wide range of design solutions, though of varying quality. More generally, Hoang et al. [13] provide a set of recommendations on when to crowdsource decision-making, namely, when tasks can be performed through the internet, do not require a significant level of communication, and can be partitioned into smaller tasks. Moreover, to avoid low quality outcomes, crowdsourced tasks should be easy to evaluate; in our case, live quality checks can be performed by both modelling and domain experts. Finally, platform availability is also key in crowdsourced tasks; our proposal based on social networks completely fulfils this requirement.

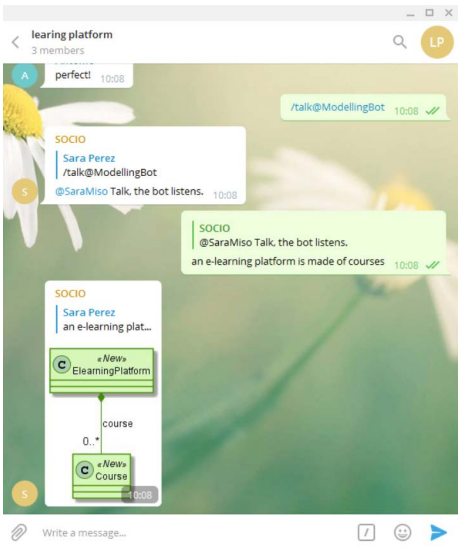
Altogether, to the best of our knowledge, our proposal is novel as modelling assistant bots have not been proposed up to now.

V. Conclusions and future work

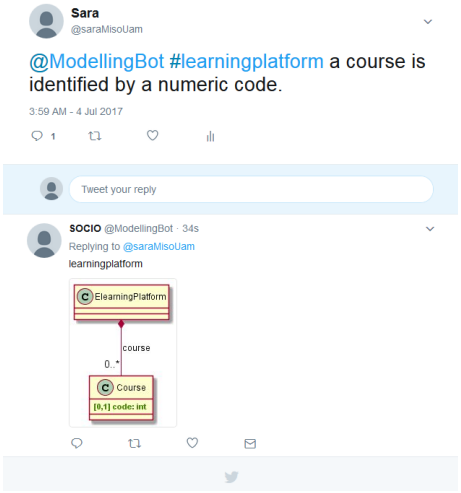
This paper has described Socio, a bot for assisted modelling over social networks. It works over Twitter and Telegram,



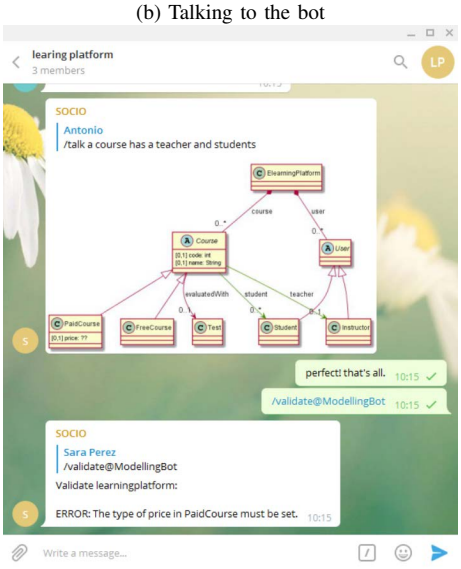
(a) Creating a new project



(b) Talking to the bot



(c) Interaction through Twitter



(d) Meta-model validation

Fig. 3: Interaction with SOCIO

interpreting NL sentences from different users to create a meta-model in a collaborative way. The assistant seamlessly integrates within the social network and keeps track of the model history and user contributions.

We are currently improving some aspects of SOCIO, e.g., enhancing NL processing for more accurate deduction of cardinalities, enabling meta-model instantiation, and defining user roles with support for collaboration protocols. We also plan to integrate other social networks and communication mechanisms, like speech recognition using Skype bots.

Acknowledgements. Work funded by the Spanish MINECO (TIN2014-52129-R) and the R&D programme of Madrid (S2013/ICE-3006).

REFERENCES

[1] M. D. Storey, A. Zagalsky, F. M. F. Filho, L. Singer, and D. M. Germán, "How social and communication channels shape and challenge a participatory culture in software development," *IEEE Trans. Software Eng.*, vol. 43, no. 2, pp. 185–204, 2017.

[2] J. Whitehead, I. Mistrík, J. Grundy, and A. van der Hoek, "Collaborative software engineering: Concepts and techniques," in *Collab. Soft. Eng.* Springer, 2010, pp. 1–30.

[3] M. D. Storey and A. Zagalsky, "Disrupting developer productivity one bot at a time," in *FSE*. ACM, 2016, pp. 928–931.

[4] S. Pérez-Soler, E. Guerra, J. de Lara, and F. Jurado, "The rise of the (modelling) bots: Towards assisted modelling via social networks," in *ASE*. IEEE, 2017.

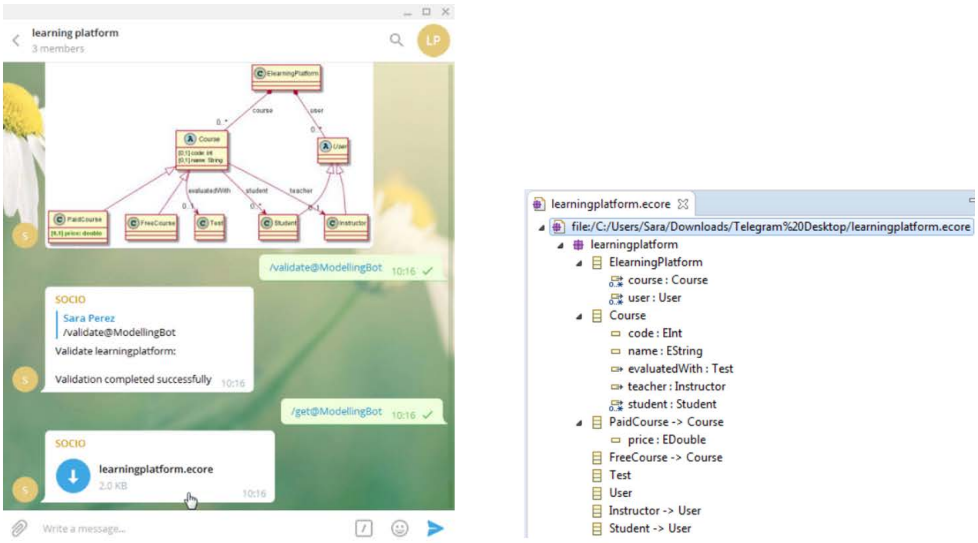
[5] M. Marneffe, B. Maccartney, and C. Manning, "Generating typed dependency parses from phrase structure parses," in *LREC*, 2006.

[6] G. A. Miller, "Wordnet: A lexical database for english," *Comm. ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[7] C. Arora, M. Sabetzadeh, L. C. Briand, and F. Zimmer, "Extracting domain models from natural-language requirements: Approach and industrial evaluation," in *MoDELS*. ACM, 2016, pp. 250–260.

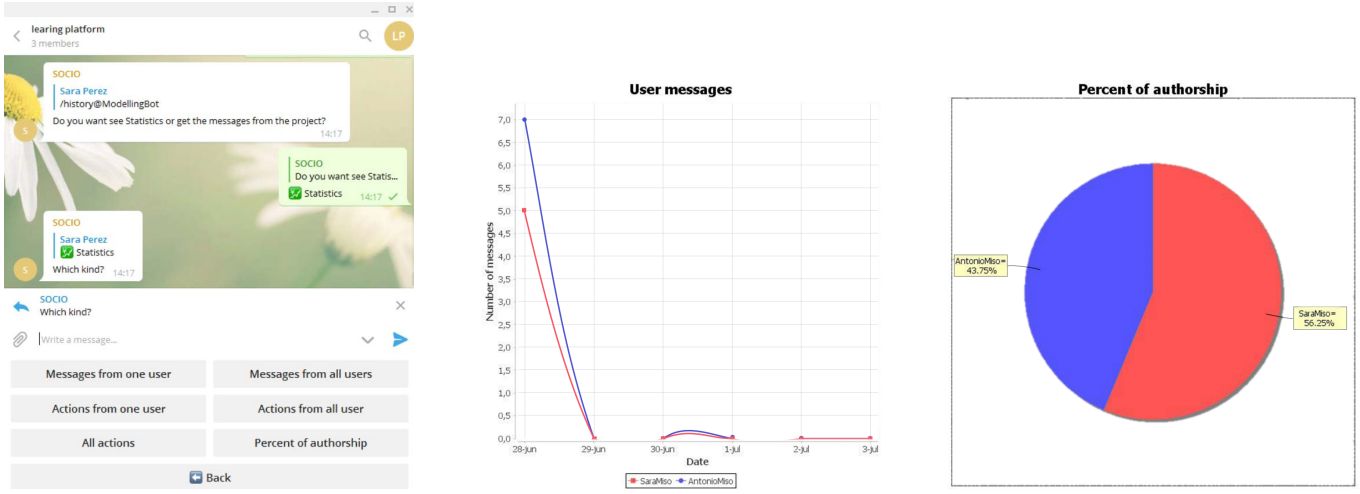
[8] M. Renger, G. L. Kolfschoten, and G. de Vreede, "Challenges in collaborative modelling: A literature review and research agenda," *IJSPM*, vol. 4, no. 3/4, pp. 248–263, 2008.

[9] J. L. C. Izquierdo and J. Cabot, "Collaboro: A collaborative (meta)



(a) Downloading Ecore meta-model (b) Ecore meta-model in Eclipse

Fig. 4: Obtaining the final meta-model



(a) Selecting statistics (b) Messages sent by users (c) Percentage of authorship

Fig. 5: Some process statistics

modeling tool," *PeerJ Computer Science*, vol. 2, p. e84, 2016.

[10] J. Gallardo, C. Bravo, and M. A. Redondo, "A model-driven development method for collaborative modeling tools," *J. Network and Computer Applications*, vol. 35, no. 3, pp. 1086–1105, 2012.

[11] A. Begel, R. DeLine, and T. Zimmermann, "Social media for software engineering," in *FoSER@FSE*. ACM, 2010, pp. 33–38.

[12] E. Weidema, C. Lopez, S. Nayebaziz, F. Spanghero, and A. van der Hoek, "Toward microtask crowdsourcing software design work," in *CSI-SE@ICSE*, 2016, pp. 41–44.

[13] N. H. Thuan, P. Antunes, and D. Johnstone, "Factors influencing the decision to crowdsource: A systematic literature review," *Information Systems Frontiers*, vol. 18, no. 1, pp. 47–68, 2016.