The Model Judge – A Tool for Supporting Novices in Learning Process Modeling

Luis Delicado[0000−0002−8866−9440], Josep Sànchez-Ferreres[0000−0002−7204−4307], Josep Carmona[0000−0001−9656−254X], and Lluís Padró[0000−0003−4738−5019]

Computer Science Department
Universitat Politècnica de Catalunya
Barcelona, Spain
{ldelicado,jsanchezf,jcarmona,padro}@cs.upc.edu

Abstract. Process models are a fundamental element in the BPM lifecycle. Hence, it is of paramount importance for organizations to rely on high-quality, accurate and up-to-date process models, to avoid taking decisions on the basis of a wrong picture of the reality. In this demo we present modeljudge.cs.upc.edu, a platform to boost the training of novice modelers when confronted with the task of translating a textual description into a process model in BPMN notation. The platform is integrated with Natural Language Processing (NLP) analysis and textual annotation, together with a novel model-to-text alignment technique. By using this platform, a novice modeler will receive diagnostics in real-time, which may contribute to a more satisfactory modeling experience.

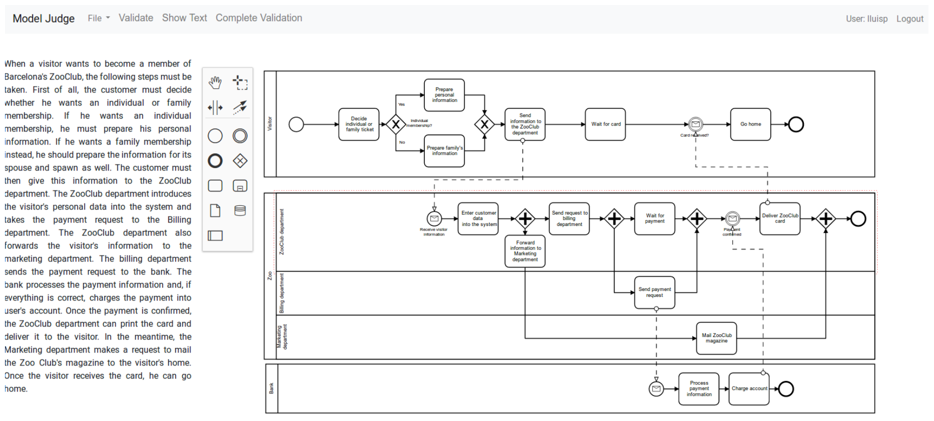
Keywords: Process Modeling · Natural Language Processing · Education

1 Introduction

Due to the wide usage of process models in organizations, correctness and quality of models have a direct influence in the execution of business processes. However, research has shown that industrial process models often contain errors, which can lead to many problems, like increased costs in production.

Automating the detection of syntactic errors is a common feature in modeling software. However, the error types more closely related to the natural language sections of the model are usually not checked, due to the difficulties in the automatic analysis of such elements. Model Judge is a web platform supporting students in the creation of business process models by automatically detecting and reporting the most common sources of semantic and pragmatic errors in modeling. The algorithm for the computation of diagnostics is based on the technique for automatic computation of alignments between process model and textual descriptions presented in [6].

Significance of the tool for the BPM field. As it is pointed out in [1], process models play a central role in the management of processes within organizations. Although recent automated techniques can help into the discovery of a process model [8], the process of process modeling it is still a crucial element in the BPM lifecycle [5]. Frameworks integrating different modeling notations, like the one presented in this paper for



When a visitor wants to become a member of Barcelona's ZooClub, the following steps must be taken. First of all, the customer must decide whether he wants an individual or family membership. If he wants an individual membership, he must prepare his personal information. If he wants a family membership instead, he should prepare the information for its spouse and spawn as well. The customer must then give this information to the ZooClub department. The ZooClub department introduces the visitor's personal data into the system and takes the payment request to the Billing department. The ZooClub department also forwards the visitor's information to the marketing department. The billing department sends the payment request to the bank. The bank processes the payment information and, if everything is correct, charges the payment into user's account. Once the payment is confirmed, the ZooClub department can print the card and deliver it to the visitor. In the meantime, the Marketing department makes a request to mail the Zoo Club's magazine to the visitor's home. Once the visitor receives the card, he can go home.

Fig. 1. Screenshot of the modeljudge workspace: textual description and model editor.

textual descriptions and BPMN, will help into narrowing the gap between processes and their representations within organizations.

2 Tool Description and Features

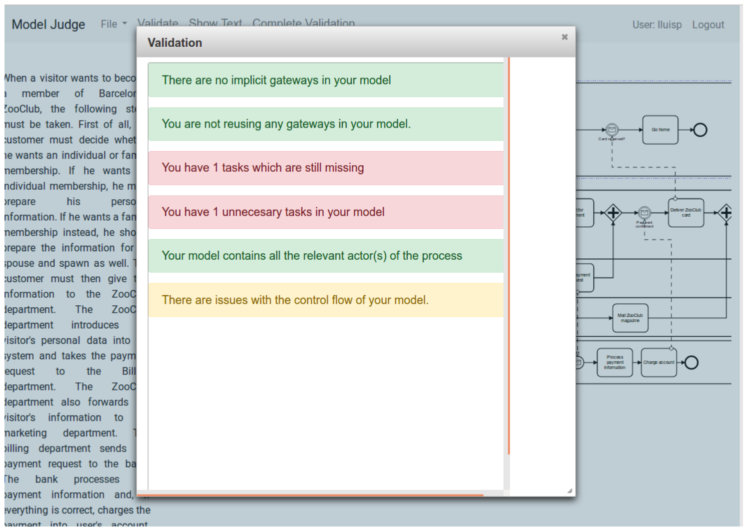
The Model Judge is presented as a web-based platform, that can be accessed through any web browser at `http://modeljudge.cs.upc.edu`. It is designed both for helping students in the process of creating a process model and instructors in the task of designing modeling activities in an agile way.

Working with the Model Judge is very much like creating a process model using any BPMN editor. What makes the platform special is the underlying engine to provide automatic feedback (see below). When modeling, a textual description of the process model is available on the left part, while the modeling editor is shown in the right part. Fig. 1 shows the workspace.

With the goal of providing an accurate evaluation of students' models, and inspired by the success of judges to support learning to program ([2,4] among many others), we have established a set of diagnostics that are suitable for being computed automatically. We have split these diagnostics in three different categories: *Syntactic* diagnostics consider the model well-formedness and control flow. *Pragmatic* diagnostics verify the phrasing of the process model labels and enforce certain grammatical rules. Finally, *semantic* diagnostics check for coverage (there is no missing information from the underlying process) and for relevance (no irrelevant information is included in the model). Fig. 2 provides different types of diagnostics reported for a particular example.

Two different types of feedback can be provided to the student, depending on the granularity of the information required:

– ***Validation***: returns an aggregated diagnostic that reflects if the model has some errors of the types explained before, but it does not say what or where is exactly



There are no implicit gateways in your model

You are not reusing any gateways in your model.

You have 1 tasks which are still missing

You have 1 unnecesary tasks in your model

Your model contains all the relevant actor(s) of the process

There are issues with the control flow of your model.

When a visitor wants to beco
a member of Barcelor
ZooClub, the following st
must be taken. First of all,
customer must decide whet
he wants an individual or fan
membership. If he wants
individual membership, he m
prepare his perso
information. If he wants a fan
membership instead, he sho
prepare the information for
spouse and spawn as well. T
customer must then give t
information to the ZooC
department. The ZooC
department introduces
visitor's personal data into
system and takes the paym
request to the Bill
department. The ZooC
department also forwards
visitor's information to
marketing department. T
billing department sends
payment request to the ba
The bank processes
payment information and,
everything is correct, charges the
payment into user's account

Fig. 2. Screenshot of the feedback for a model where a relevant task *introduce user data into the system* has been replaced by a non-relevant task *feed the dog*.

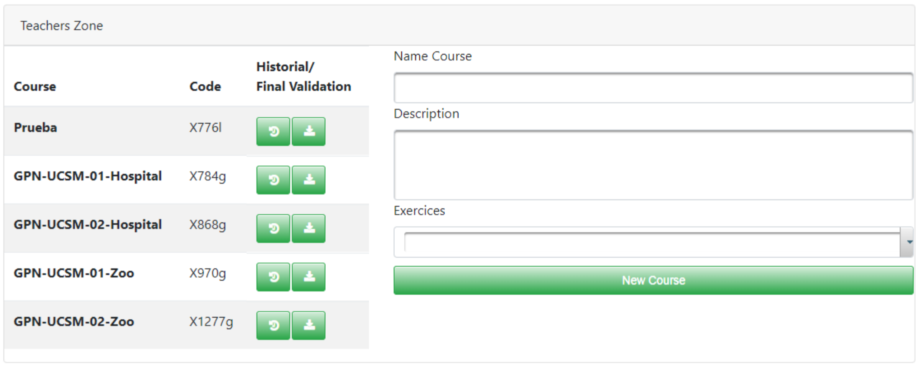
the problem. The motivation for this check is to allow for a mild test to guide the students without giving away the whole solution. The feedback provided in Fig. 2 is a validation.

- ***Complete Validation***: apart from the overall information provided by the Validation, this check also provides a detailed list of all the problems detected, and individually explained. The motivation for this check is to allow an assessment similar to the one obtained if a teacher was correcting the model, and would be typically provided once the student finishes and hands over the exercise.

In order to use the platform, a user (either student or instructor) needs an account, with its associated storage space. This space acts as a cloud drive for models, allowing to manually save multiple versions of an exercise. Moreover, the students enrolled in a particular course, may enable the platform to record a history of their modeling session. This can be used for analyzing their behavior, which can be reported back to the instructor to get a clearer picture of the modeling process of their students.

Currently, there are 9 different exercises available in the platform. Any modeler can register into the platform and practice with them. Instructors can also design a course, by selecting the exercises that must be included. Fig. 3 shows the main page for defining a course as instructor. Once a course is created, a unique code will be created that can be shared with the students of this course.

Support for adding new exercises is restricted to the developers of the platform. However, support is planned to allow instructors to create their own exercises. In order to create a new exercise for the Model Judge, a textual description of a process is required. This textual description will be used as the problem statement for the students so they can understand the process to be modeled. Instructors will then have to anno-



Course	Code	Historial/ Final Validation
Prueba	X776l	
GPN-UCSM-01-Hospital	X784g	
GPN-UCSM-02-Hospital	X868g	
GPN-UCSM-01-Zoo	X970g	
GPN-UCSM-02-Zoo	X1277g	

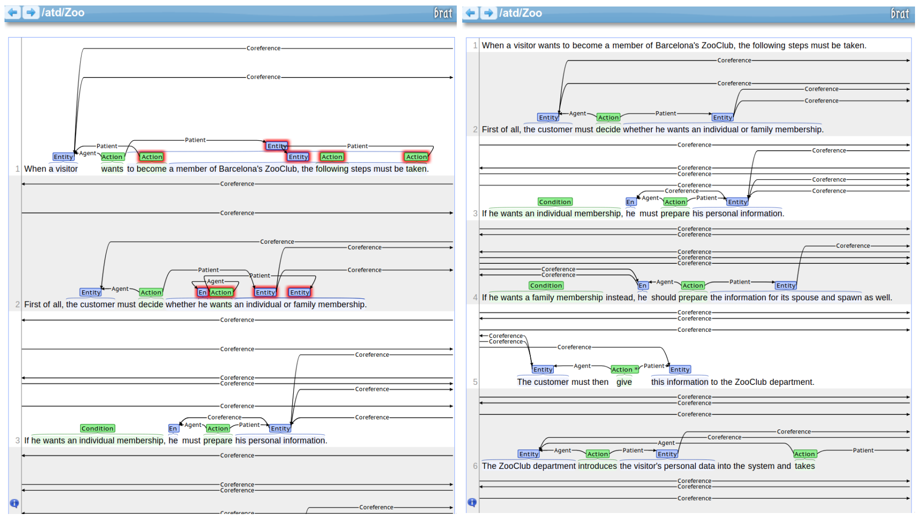
Name Course

Description

Exercices

Fig. 3. Screenshot of the interface for the definition new courses by an instructor.

tate the relevant parts of the process: *Actions*, *Entities* and *Conditions* as well as their relations: *Agent*, *Patient*. This annotation process is partially performed by a Natural Language Processing algorithm, which provides an initial annotation to be refined. Figure 4 shows a fragment of a text annotation corresponding to one of the exercises in the platform.



1 When a visitor wants to become a member of Barcelona's ZooClub, the following steps must be taken.
2 First of all, the customer must decide whether he wants an individual or family membership.
3 If he wants an individual membership, he must prepare his personal information.

1 When a visitor wants to become a member of Barcelona's ZooClub, the following steps must be taken.
2 First of all, the customer must decide whether he wants an individual or family membership.
3 If he wants an individual membership, he must prepare his personal information.
4 If he wants a family membership instead, he should prepare the information for its spouse and spawn as well.
5 The customer must then give this information to the ZooClub department.
6 The ZooClub department introduces the visitor's personal data into the system and takes

Fig. 4. The annotation interface for the definition of exercises: (left) An automatically generated annotation. (right) After the instructor performs a manual validation.

A screencast of the Model Judge which shows a typical session working with the model judge can be found in `https://youtu.be/xJ3TeKlvIfo`.

3 Architechture, Libraries Used and Maturity of the Tool

Model Judge is built as a web application. A distributed server manages several instances of the application and balances the load between them. The front-end of it is built using the PrimeFaces Java framework, which internally communicates with the core Java application, responsible for the generation of diagnostics. Several functionalities in Model Judge rely on external libraries: *FreeLing* [3] is used for all the Natural Language Processing tasks and the *Gurobi* ILP solver is used to compute optimal alignments, as described in [6]. Additionally, the BRAT [7] text annotation software is used as an external tool in order to create new exercises for the platform.

The Model Judge has been tested in two separate modeling courses. The first was performed on the Technical University of Denmark (DTU) during February 2018. The second course was performed at the Catholic University of Santa María (UCSM) in Peru during March 2018. For every student of these courses, we stored periodically (every minute) information for the whole modeling session. Additionally, information was also saved each time the user performed a simple or complete validation. In particular, we recorded a total of 8410 intermediate models for 72 students.

References

1. Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management*. Springer, 2013.
2. Adrian Kosowski, Michal Malafiejski, and Tomasz Noinski. Application of an online judge & contester system in academic tuition. In *Advances in Web Based Learning - ICWL 2007, 6th International Conference, Edinburgh, UK, August 15-17, 2007, Revised Papers*, pages 343–354, 2007.
3. Lluís Padró and Evgeny Stanilovsky. Freeling 3.0: Towards wider multilinguality. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC*, pages 2473–2479, Istanbul, Turkey, May 2012.
4. Jordi Petit, Omer Giménez, and Salvador Roura. Jutge.org: An educational programming judge. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, SIGCSE '12, pages 445–450, New York, NY, USA, 2012. ACM.
5. Jakob Pinggera, Pnina Soffer, Dirk Fahland, Matthias Weidlich, Stefan Zugal, Barbara Weber, Hajo A. Reijers, and Jan Mendling. Styles in business process modeling: an exploration and a model. *Software and System Modeling*, 14(3):1055–1080, 2015.
6. Josep Sànchez-Ferreres, Josep Carmona, and Lluís Padró. Aligning textual and graphical descriptions of processes through ILP techniques. In *Advanced Information Systems Engineering - 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings*, pages 413–427, 2017.
7. Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics, 2012.
8. Wil M. P. van der Aalst. *Process Mining - Data Science in Action, Second Edition*. Springer, 2016.