# TAGRAM: A Framework for Tagging User Stories

Maxim Bragilovski, Shahaf Erez, Chen Mordehai, Shani Rahamim, Noa Shpack, Arnon Sturm

***Abstract*—User stories are a popular notation for representing requirements, especially in agile development. However, how they can be used further during the development process is still under examination. In recent years, user stories research has focused on how to apply machine learning to user stories to understand their quality, the required implementation efforts, and the ability to generate software artifacts. One primary concern in this direction is the lack of datasets and specially labeled ones. To bridge this gap, in this work, we introduce a framework for tagging user stories to construct labeled datasets for user stories for various tasks.**

## I. Introduction

User stories are a popular technique for expressing requirements [4]. Through their simple notation, they represent the stakeholder, the feature requested, and the rationale behind the feature. The so-called Connextra notation [4] "*As a* ⟨*role*⟩ *I want to* ⟨*feature*⟩ *so that* ⟨*benefit*⟩" is widely used for the representation of the elicited requirements in agile development projects [8], [11]. User stories play a central role in the subsequent stages of software development [12], [1]. They may be *refined* into lower-level specifications, such as conceptual models [6], [5], used to estimate the required development efforts and applied to manage projects.

In parallel with the development of the usage and importance of user stories, artificial intelligence (AI) and, in particular, machine learning (ML) is beginning to be applied to user stories to check their quality, for example [9], estimate the required efforts, for example [3], and generate new artifacts [2]. However, this research lacks datasets that can be used for learning.

To address this gap, in this work, we propose TAGARM[1] (a composition of tagging and grammar) - a framework that can support the preparation of such labeled datasets. This is where the core artifacts are the user stories, and these will be equipped with additional information - the required tagging. In the following, we elaborate on the framework's functionality and architecture.

## II. TAGRAM Functionality

TAGRAM is organized along the conceptual model in Figure 1. A project is an anchor around which the framework is organized. A single tagging scheme is established within a project, dictating the appropriate guidelines for tagging the user stories associated with that particular project. We allow a project to be associated with only one scheme that is aligned with the tagging goal. In case multiple goals are required from a specific dataset, multiple projects should be defined. Additionally, the project retains a comprehensive record of the tagging (the Tagging class) carried out by the assigned human taggers and automatic ones. The following section outlines the diverse functionalities supported by the framework.

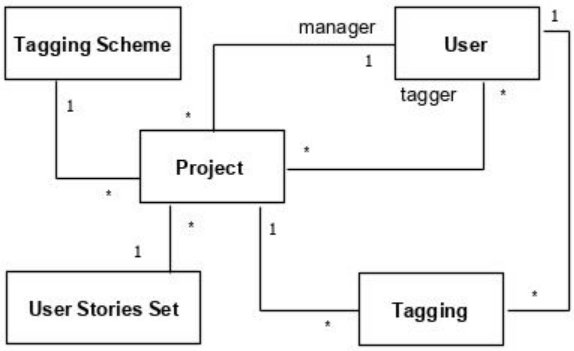


Fig. 1: TAGRAM Conceptual Model

- **Manage Users** The users of the framework include three types: *Administrators* have the ability to manage user accounts and permissions. *Project Managers* have the ability to define tagging projects, including the definition of user stories, the tagging scheme, and the management of taggers. *Taggers* can only tag user stories without accessing each other taggings. This functionality ensures that access to user stories and other project-related information is controlled and restricted to authorized users.
- **Manage Projects** To start a tagging task, a project manager should define a project, a set of user stories, and a related tagging scheme (which can be defined within the framework). Later, she can manage the taggers for that project. The framework facilitates the management of multiple projects simultaneously. Figure 2 shows how to define a project.
- **Define Tagging Scheme** To start the tagging process, there is a need to specify the required labels and tags. In TAGRAM we adopted a graph approach in which we define entities (nodes) and binary relationships (edges) among them. We label each of them and assign a color to that label to ease the tagging process. See Figure 4.
- **Load User Stories** The framework enables project managers to import user stories from text files. This feature simplifies the tagging process as users can work on an already provided set of user stories, thus saving time and effort.
- **Check the Quality of User Stories** The framework allows managers to assess the quality of user stories based on predefined criteria. It helps identify common

[1]https://tagram.cs.bgu.ac.il/

issues like ambiguity, incompleteness, or conflicting requirements. We have integrated the AQUSA tool [10] for that task.

- **Update Project** Tagging projects can be easily modified and updated by project managers. This includes various modifications, such as assigning new taggers and adjusting the user stories linked to the project. The ability to modify user stories allows project managers to enhance existing user stories and introduce new ones before undertaking the tagging task. Such adaptability ensures that the tagging projects remain dynamic and responsive to evolving requirements and improvements. See Figure 3.
- **Manual Tagging of User Stories** Upon defining the tagging scheme, taggers can assign the labels to part of the text of the user story. The tagging appears in a list, so taggers can track the process and verify their outcomes. See Figure 5.
- **Automatic Tagging of User Stories** Similar to manual tagging, TAGRAM supports automatic tagging by applying predefined algorithms. The algorithms should follow the tagging scheme (or a subset of that) assigned to the project. Currently, we have implemented one such algorithm. However, this can be replaced by others.
- **Calculate Tagging Statistics** TAGRAM provides statistical analysis and reporting on the tagging outcomes. It generates insights into tag frequencies and tagger inter agreement using Fleiss Kappa [7].
- **Export Tagging Results** Project managers can export the tagging results for further analysis and manage their repositories (e.g., at the organization infrastructure).
- **Assign Tagging Algorithm** TAGRAM allows project managers to load and select tagging algorithms. This flexibility enables customization based on specific project requirements or goals.

The demonstration includes the functionality described above. We start by playing the role of a project manager to specify a project along its goals by defining the tagging scheme, loading user stories, and assigning taggers. Next, we move to the tagging activities, playing the role of a tagger. Finally, we return to the project manager role to activate the automatic tagging, check the agreement among taggers, and export the results.

## III. TAGRAM Architecture

TAGRAM is implemented as a web application built on top of the React.js and Django frameworks. The system's architecture follows a client-server model, with React being the front end handling the user interface and Django being the back end, managing data storage and business logic.

The front end of the system presents an intuitive and interactive interface designed to enhance user experience. This user-friendly interface is designed for various user roles within the system, including project managers, taggers, and system administrators. Each user category is granted specific permissions and capabilities, ensuring efficient utilization of the system. The interface facilitates seamless interaction and navigation, enabling users to effectively perform their tasks based on their designated roles and responsibilities.

The back end of the system is responsible for managing server-side logic and data operations. It efficiently handles the storage of user-related information, including user profiles, project data, and taggings. Moreover, it provides a convenient feature that allows users to save incomplete work and resume it later.

To facilitate the annotation functionality for the tagging task, TAGRAM incorporates the use of the React-Text-Annotation library. This library enhances the capabilities of TAGRAM by providing a comprehensive and efficient solution for annotating text within the system. Using the React-Text-Annotation library, TARGRAM allows tagging user stories with a selected tagging scheme.

Overall, TAGRAM's implementation as a web application powered by React.js and Django, delivers a user-centric interface, data management, and seamless annotation functionality.

## IV. Preliminary Evaluation

Nowadays, we are evaluating the tool's usability by means of a focus group working with the tool to prepare datasets for identifying entities and relationships. The preliminary results indicate that the participants found that the tool was useful for annotating user stories for the purpose of identifying entities and relationships. Using the tool we also checked the inter-agreement among the annotators and executed the automated algorithm.

## References


[1] Jasper Berends and Fabiano Dalpiaz. Refining User Stories via Example Mapping: An Empirical Investigation. In *Proc. of RE, Industrial Innovation Track*, 2021.

[2] M. Bragilovski, F. Dalpiaz, and A. Sturm. From user stories to domain models: Recommending relationships between entities. In *CEUR Workshop Proceedings*, volume 3378, 2023.

[3] Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, Trang Pham, Aditya Ghose, and Tim Menzies. A deep learning model for estimating story points. *IEEE Transactions on Software Engineering*, 45(7):637–656, 2019.

[4] Mike Cohn. *User Stories Applied: for Agile Software Development*. Addison Wesley, 2004.

[5] Fabiano Dalpiaz, Patrizia Gieske, and Arnon Sturm. On deriving conceptual models from user requirements: An empirical study. *Inf. Softw. Technol.*, 131:106484, 2021.

[6] Fabiano Dalpiaz and Arnon Sturm. Conceptualizing requirements using user stories and use cases: A controlled experiment. In *Proc. of REFSQ*, pages 221–238, 2020.

[7] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[8] Mohamad Kassab. An Empirical Study on the Requirements Engineering Practices for Agile Software Development. In *Proc. of EUROMICRO SEAA*, pages 254–261, 2014.

[9] Yarden Levy, Roni Stern, Arnon Sturm, Argaman Mordoch, and Yuval Bitan. An impact-driven approach to predict user stories instability. *Requir. Eng.*, 27(2):231–248, jun 2022.

[10] G. Lucassen, F. Dalpiaz, J.M.E.M. van der Werf, and S. Brinkkemper. Improving Agile Requirements: The Quality User Story Framework and Tool. *Requir. Eng.*, 21(3):383–403, 2016.

[11] G. Lucassen, F. Dalpiaz, J.M.E.M. van der Werf, and S. Brinkkemper. The Use and Effectiveness of User Stories in Practice. In *Proc. of REFSQ*, pages 205–222, 2016.

[12] Laurens Müter, Tejaswini Deoskar, Max Mathijssen, Sjaak Brinkkemper, and Fabiano Dalpiaz. Refinement of user stories into backlog items: Linguistic structure and action verbs. In *Proc. of REFSQ*, pages 109–116, 2019.

## V. Appendix

The appendix presents selected screenshots of TAGRAM.

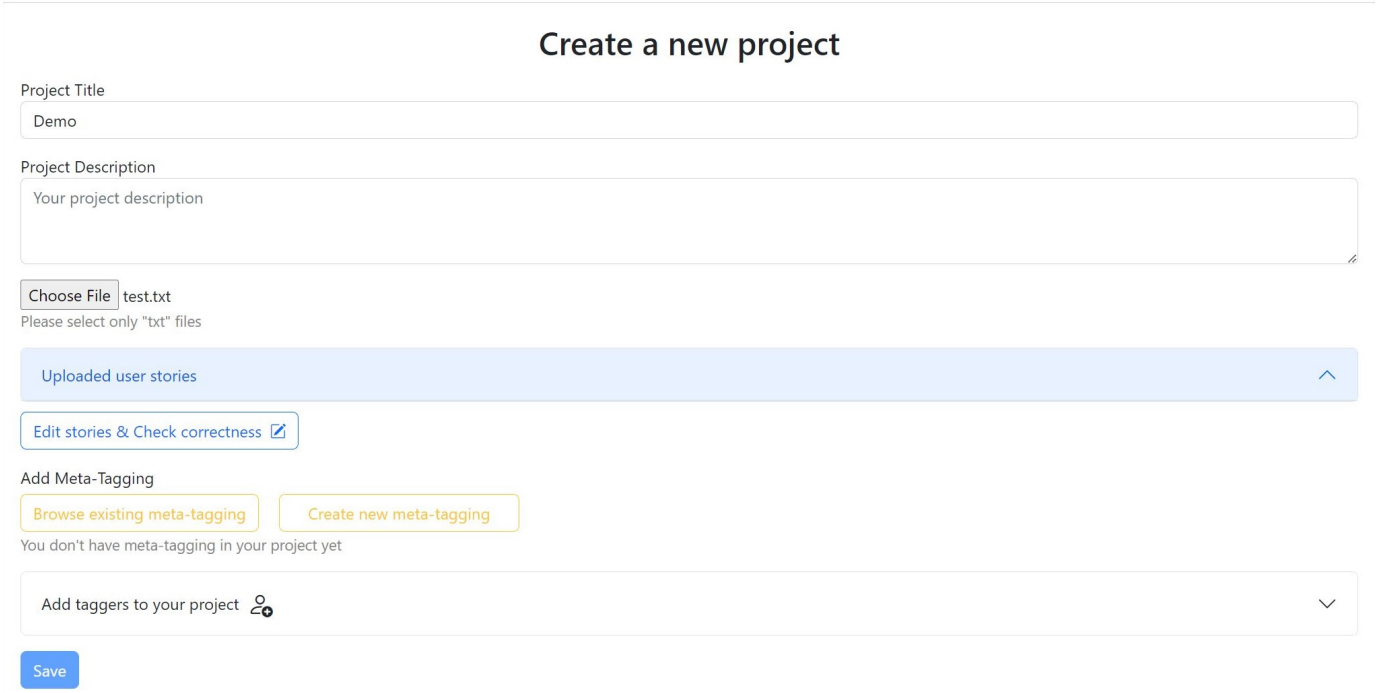


Fig. 2: Create a new project

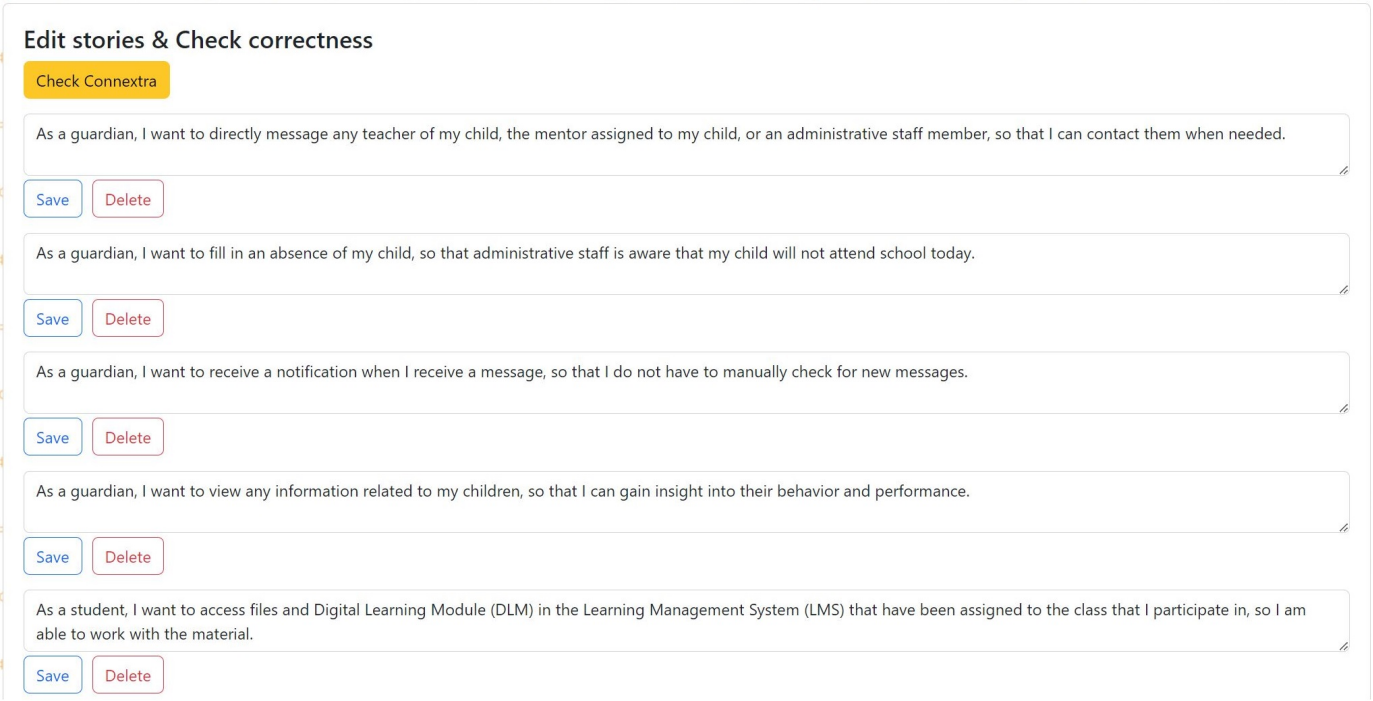


Fig. 3: Edit and Check User Stories

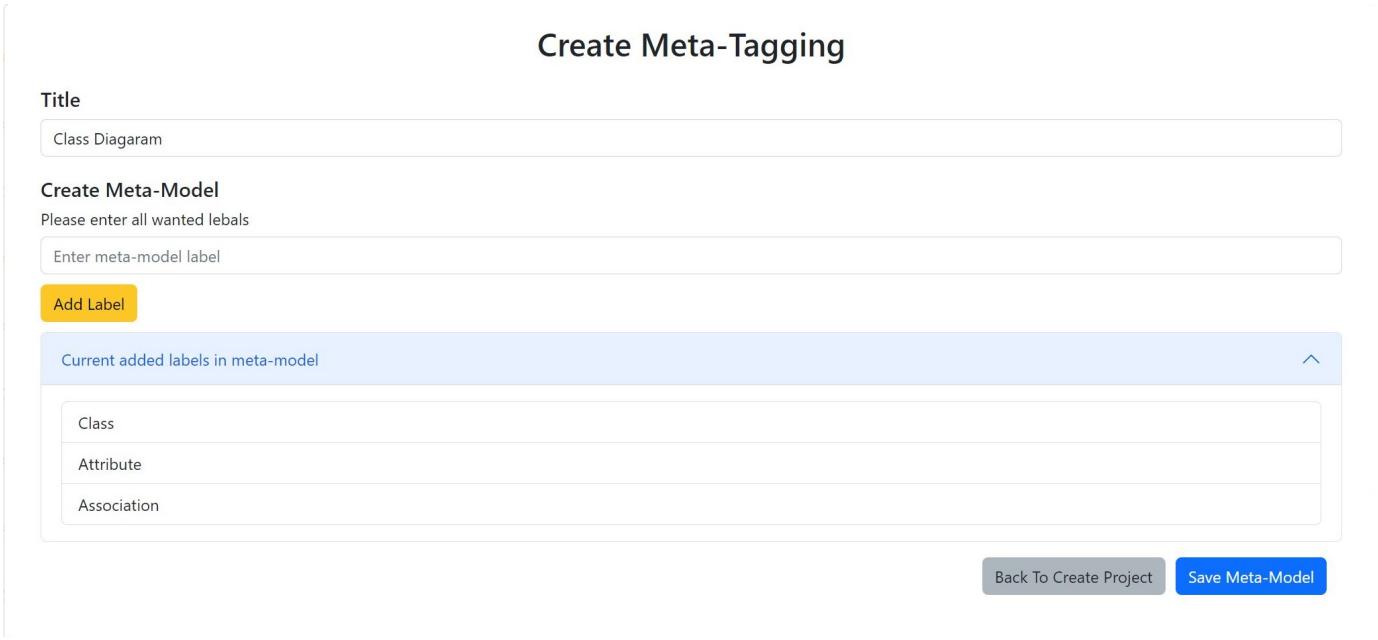


Fig. 4: Meta Tagging Label Definition

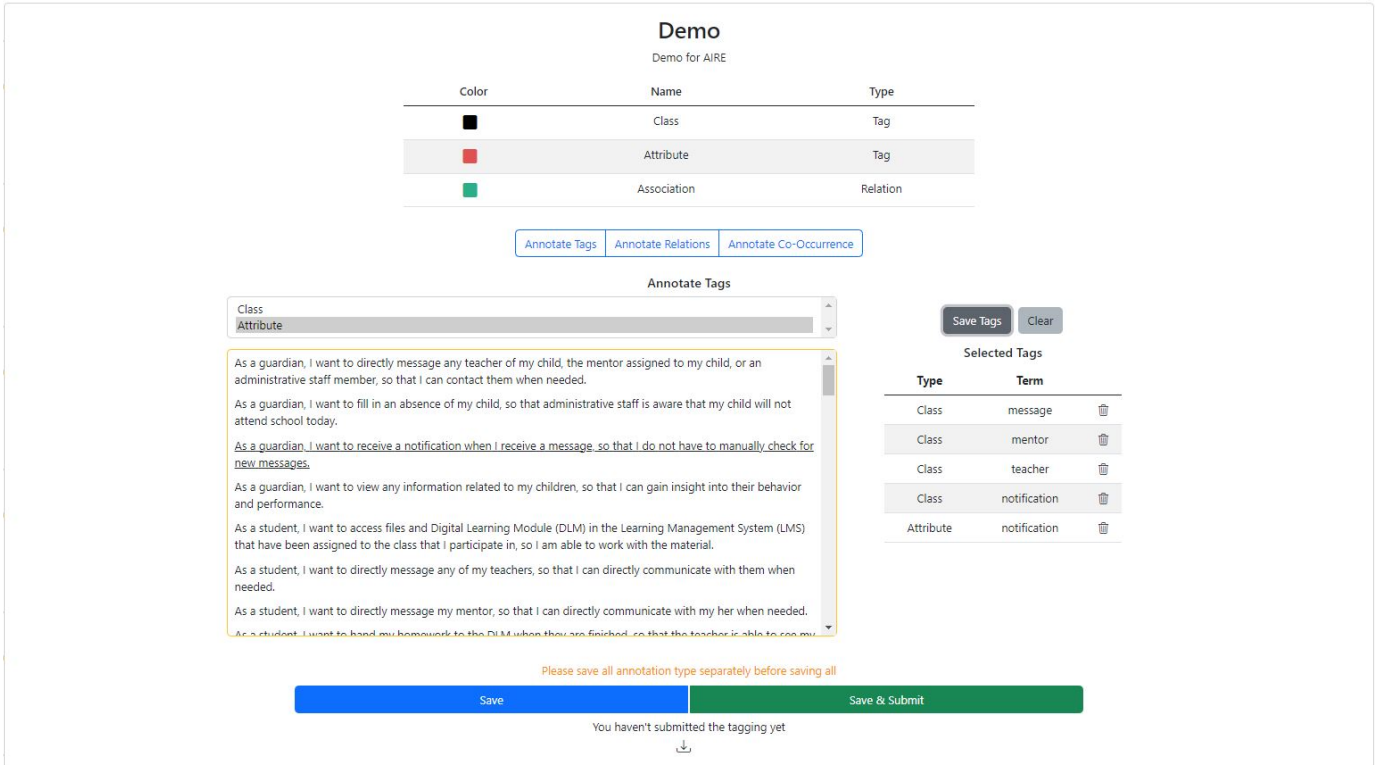


Fig. 5: The Annotation Page