Text2Stories : Extraction et Génération de *User Stories* à partir de Documentation Fonctionnelle et Non-Fonctionnelle

Marius Ortega [*,**], Hassan Imhah [*], Vanande Khatchatrian [*]
Nédra Mellouli [**], Christophe Rodrigues [**], Nicolas Travers [**]

[*]Onepoint, 29 rue des Sablons, 75116, Paris
première-lettre-du-prénom.nom@groupeonepoint.com
[**]Pole Léonard de Vinci, 92 916, Paris, La Défense
prénom.nom@devinci.fr,

Résumé. Les *user stories* (US) sont un outil clé en ingénierie logicielle. Hérité des méthodes Agiles, il permet d'exprimer des exigences fonctionnelles et non-fonctionnelles du point de vue de l'utilisateur final. Leur étude en traitement automatique du langage naturel présente toutefois plusieurs limitations. Les méthodologies Agile et open source sont partiellement incompatibles : la première suppose une collaboration étroite et co-localisée, tandis que la seconde repose plus fréquemment sur des équipes distribuées et asynchrones. Il en résulte peu de jeux de données publics, et aucun ne combinant des US de haute qualité, métriques établies et éléments contextuels du projet. En pratique, les US sont souvent mal formulées et évaluées au travers de révisions informelles et orales telles que la cérémonie des « *Trois Amigos* ». Ces limites freinent la progression des méthodes de génération et évaluation automatique d'US. Cet article présente un outil d'IA Multi-Agents générant et évaluant des US contextualisées, fournissant des résultats comparables et reproductibles pour chercheurs et praticiens.

1 Introduction

Les *user stories* (US) se sont imposées comme un standard pour transcrire des exigences fonctionnelles et non fonctionnelles du point de vue de l'utilisateur (R. Amna et Poels, 2021). Ce sont de court textes rédigés en langage naturel et hérités du *Expert Programming* (Wake, 2003). Leur structure la plus courante se compose de trois éléments : un Acteur qui est un utilisateur final du produit, une Action effectuée par cet Acteur et une Valeur décrivant l'avantage qu'apporte la réalisation de cette Action pour l'utilisateur. Elles s'écrivent : « En tant que <Acteur>, je veux <Action>, afin de <Valeur> ». Les US peuvent contenir des images et des critères d'acceptation permettant de tester la qualité de leur implémentation. Leur valeur n'émerge qu'une fois contextualisées dans des projets, puisque le sens qu'elles portent individuellement est peu informatif sur la totalité du projet. Elles sont généralement regroupées en blocs de fonctionnalités appelés *Epics* et stockées dans le backlog produit avant leur implémentation. Bien que l'usage des US soit devenu incontournable, leur utilisation dans les contextes industriels ainsi qu'académiques souffre de limitations systémiques et techniques.

Les limitations systémiques sont inhérentes aux domaines du Requirement et du Software Engineering (RE/SE). Parmi celles-ci, on trouve l'incompatibilité partielle entre les méthodologies open source et Agile. La première suppose une collaboration étroite et co-localisée, tandis que la seconde repose sur des équipes distribuées et asynchrones. Par conséquent, la plupart des projets Agile sont privés et inaccessibles dans le cadre de la recherche ouverte. Il résulte peu de jeux de données publics accessibles répertoriant des US (Dalpiaz et al., 2019; Neo et al., 2024; Tawosi et al., 2022) et aucun ne fournit à la fois des artefacts de haute qualité, avec des métriques établies (INVEST (Wake, 2003), QUS (Lucassen et al., 2016)) et une contextualisation relative au projet (*epic stories*, critères d'acceptation). Les limitations techniques sont spécifiques à notre cas d'usage qui consiste en la rédaction et l'évaluation automatiques d'US. Il ressort que les US sont fréquemment mal rédigées et sont généralement évaluées de manière informelle et non systématique ; par exemple lors de la cérémonie Agile des « *Trois Amigos* ».

Ces limitations entraînent une pénurie de données ralentissant les progrès du NLP dans ce domaine (Raharjana et al., 2021). En particulier, l'évaluation de la qualité des US (Lucassen et al., 2016; Ronanki et al., 2024) et leur génération (Rahman et al., 2024; Yamani et al., 2025) restent des tâches difficiles. Enfin, à notre connaissance, aucune solution ne traite à la fois la génération d'US contextualisées, de critères d'acceptation et l'évaluation de la qualité avec des métriques standard (Rahman et al., 2024; Yamani et al., 2025).

Par conséquent, nous présentons Text2Stories (`T2S`), un outil conçu pour assister praticiens et chercheurs dans les tâches de génération et d'évaluation automatique d'US à partir de descriptions en langage naturel. Notre agent IA fait notamment usage des *Chain-of-Thoughts* (CoT) (Wei et al., 2022) pour améliorer la qualité de raisonnement des Large Language Models (LLM), la communication normée entre sous-agents (Schluntz et Zhang, 2024) pour l'explicabilité et la création de sous-objectifs, des méthodes de *Retrieval-Augmented Generation* (RAG) (Lewis et al., 2020) pour l'adaptation de domaine, et enfin l'architecture d'agent ReAct (Yao et al., 2023) pour l'exploration autonome. `T2S` génère non seulement des *epic stories* et des US, mais fournit également une notation automatique s'appuyant sur l'ensemble de métriques métier le plus utilisé, INVEST[1]. Nous complétons ces évaluations automatiques par une évaluation humaine menée par 20 consultants experts du test logiciel. Notre outil (`T2S`) est disponible sous licence open-source sur notre dépôt github[2] et supporte deux types d'usagers :

— **Les *Product Owner*** (PO) pour la réécriture d'US, obtenir des propositions d'US dans un contexte donné, ou encore évaluer la qualité des éléments existants.

— **Les Chercheurs** en RE/SE et NLP afin de générer des jeux de données synthétiques pour l'entraînement, l'évaluation, la reproductibilité, ou le finetuning des LLM.

2 Text2Stories : Une Approche Agentique

`T2S` est un outil modulaire composé de 4 blocs répliquant la structure d'une équipe Agile (Fig. 1) : les blocs *Élicitation*, *Scheduling*, *Vérification* et *Dégradation*. La modularité de `T2S` présente deux avantages : éviter l'obsolescence rapide et permettre l'utilisation de tout ou partie des modules de Text2Stories avec une interchangeabilité des LLM sous-jacents. `T2S` est conçu pour recevoir une requête en input (ex : « Création d'une application pour l'accord de

1. INVEST : *Independant*, *Negotiable*, *Valuable*, *Estimable*, *Small*, *Testable*
2. Dépôt GitHub : `https://github.com/MariusAAROS/Text2Stories`

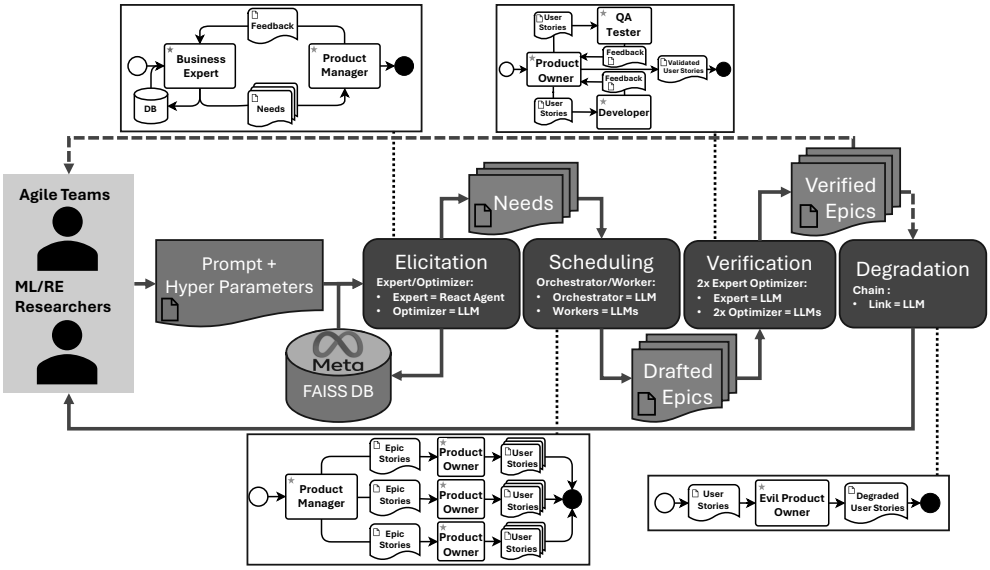

FIG. 1 – *Text2Stories Pipeline (★ : LLM, ★ : Agent ReAct)*

crédit au particulier. »). Ensuite, l'outil s'appuie sur la documentation du projet vectorisée avec FAISS (ex : documents de spécification, discussion entre acteurs du projet etc.) pour produire un document JSON composés d'US, epic, critères d'acceptation et critères INVEST.

Le *bloc d'élicitation* est basé sur un sous-agent *Expert/Optimizer* (Schluntz et Zhang, 2024) où le nœud *Business Expert* est un agent ReAct (Yao et al., 2023) lié à une base de données vectorielle FAISS, tandis que le nœud *Product Manager* est un LLM avec des instructions de prompting strictes (LLM contraint de produire un JSON). Le *Business Expert* (expert) génère des besoins en interrogeant la base de données vectorielle jusqu'à ce que le *Product Manager* (*Optimizer*) les considère comme "suffisants" pour constituer le backlog du produit.

Le *bloc de planification* exploite la parallélisation grâce à un sous-agent *Orchestrator/Worker*. Le nœud *Product Manager* (Orchestrator) rassemble les besoins créés pendant la phase d'élicitation et construit des *epic stories* (les blocs fonctionnels). Chaque *epic* est attribuée de manière asynchrone à un *Product Owner* (*Worker*) qui génère les US associées à l'*epic*. Ce sous-agent est particulièrement utile lorsque le nombre de tâches à accomplir n'est pas connu à l'avance.

Le *bloc de vérification* représente une transposition agentique du rituel des *Trois Amigos* dans les méthodologies Agile. Le *Product Owner* (PO), le *Quality Assurance Tester* (QA) et le *Developer* (DEV) représentent trois LLM, chacun spécialisé via prompting dans des critères INVEST spécifiques liés aux activités réelles des équipes Agile. Le PO est responsable des critères produit (Negotiable, Estimable, Valuable, Small), le QA se limite à la testabilité comprenant la vérification des critères d'acceptation, et enfin, le DEV se concentre sur l'indépendance, reflétant la faisabilité de l'implémentation de l'US. Du fait de l'absence de données annotées, les LLM prennent leur décision via classification *Zero-Shot* (Chae et Davidson, 2025) : méthode qui provient du domaine du *Transfer Learning*.

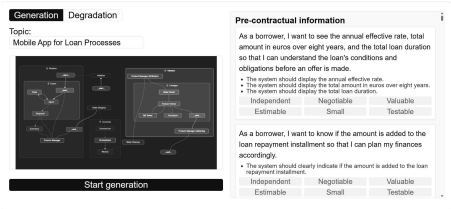

FIG. 2 – *Interface Visuelle de* `T2S` *Mode Génération*

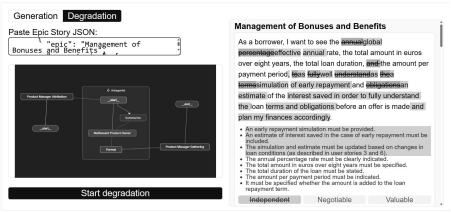

FIG. 3 – *Interface Visuelle de* `T2S` *Mode Dégradation*

Enfin, **le *bloc de dégradation*** est un LLM standard, le *Evil Product Owner* est prompté de manière souple (format de sortie libre du LLM) pour dégrader la qualité des US selon les critères INVEST souhaités. Ce dernier bloc est facultatif et est conçu pour diversifier la sortie du module de génération (élicitation, planification et vérification). Alors que le module de génération produit systématiquement des US de haute qualité, l'étape de dégradation introduit une variation contrôlée de la qualité INVEST des US, une caractéristique cruciale pour construire des benchmarks robustes. Par conséquent, le module de dégradation est principalement destiné à la recherche en Machine Learning.

3 Plan de Démonstration

Notre scénario de démonstration est divisé en trois étapes comprenant des éléments interactifs et éventuellement l'intervention du public. Notre démonstration vise à montrer l'efficacité de Text2Stories pour la génération et l'évaluation de US dans le cadre d'un projet, et non pas uniquement en tant qu'éléments isolés.

Sélection de cas d'usage Nous commençons notre démonstration en présentant et en choisissant parmi les deux cas d'usage traités par `T2S` (Figs. 2 et 3) :

— Cas d'usage 1 : Assistant au brainstorming pour les équipes agiles incluant la création complète d'un backlog à partir d'une requête en langage naturel. Les utilisateurs spécifient les hyper-paramètres du modèle dans l'interface de la fenêtre et définissent le sujet de génération dans le champ de saisie associé. `T2S` produit un fichier JSON contenant des *epic* avec US, critères d'acceptation et INVEST associés (Fig. 2).

— Cas d'usage 2 : Création d'US contextualisées illustrant comment `T2S` effectue une diversification contrôlée des exigences selon des critères INVEST spécifiques. Il prend en entrée une *user story* ou une *epic story* au format JSON et produit des exigences dégradées dans un autre fichier JSON (Fig. 3).

Une fois le cas d'usage choisi, nous pouvons choisir un sujet spécifique pour la génération, tel que "*Création d'une application mobile facilitant les processus d'approbation de prêts*".

Présentation des pipelines Nous proposons de présenter les pipelines de génération et dégradation de Text2Stories avec Langgraph Studio, qui fait partie de l'écosystème Langchain et permet de visualiser des modèles agentiques pendant leur exécution. Notre modèle actuel est présenté via une application *Next.js* avec *React* et *TypeScript*, stylisée à l'aide de *Tailwind CSS*

(Figs. 2 et 3). Au cours de la démonstration, le public peut visualiser en temps réel la structure du modèle (Fig. 2).

Exploration des résultats Enfin, nous parcourons le fichier JSON généré par `T2S` (à droite de l'écran) : *Epic*, US, critères d'acceptation et métriques INVEST pour le module de génération (Fig. 2), ainsi que la version diversifiée après dégradation (Fig. 3).

4 Évaluation

4.1 Création des Jeux de Données

Nous avons effectué trois ablations sur notre modèle afin de démontrer la pertinence de son architecture. Initialement, Text2Stories combine des sous-agents composés de LLM, RAG, et agents ReAct ainsi que des connaissances préalables en ingénierie des exigences. Nous utilisons `mistral-large-123b` comme LLM pour toutes nos expérimentations.

La première ablation est l'expérience `RAG+CoT`. Comparée à Text2Stories, elle n'intègre ni agents, ni connaissances préalables des processus (Cérémonie des *Trois Amigos* par exemple). La seconde (`LLM+CoT`) exploite un LLM au lieu d'un RAG, privant ainsi le modèle de sources de données externes spécifiques au cas d'usage. Enfin, l'expérience `LLM` ne décompose pas la création du backlog en plusieurs appels aux LLM (élicitation, planification, vérification), mais produit le backlog en un seul appel au modèle.

A l'issue de l'exécution des trois ablations et des deux modules de `T2S`, nous créons un ensemble de données $\mathcal{D}$ de 420 US (100 US par ablation et 60 US pour les modules de génération et dégradation de `T2S`). L'ensemble du processus de création des US dans ce papier s'est concentré sur le domaine de l'assurance et de l'accord de prêts car la documentation est open source. Afin de comparer nos données avec l'évaluation humaine, nous créons un sous-ensemble de données $\mathcal{D}'$ composé d'*epics* et d'US tirées aléatoirement de $\mathcal{D}$. $\mathcal{D}'$ est donc composé de 129 US, avec au moins 25 US par expérience (`LLM`, `LLM+CoT`, `RAG+CoT`, `T2S GEN` et `T2S DEG`). Notre processus d'évaluation par l'humain s'appuie sur $\mathcal{D}'$.

4.2 Dispositif Expérimental

Comme présenté dans la section 1, la génération et l'évaluation des US ont été abordées dans des travaux de recherche antérieurs (Lucassen et al., 2016; Rahman et al., 2024; Ronanki et al., 2024; Yamani et al., 2025).

Il apparaît que `T2S` recouvre les 3 aspects que nous considérons dans notre cadre expérimental : **i) la génération en contexte**, c'est-à-dire que les US ne sont pas générées seules, mais conjointement avec d'autres et dans le cadre d'un projet spécifié afin de se rapprocher de leur utilisation réelle. **ii) la génération de critères d'acceptation** pour des exigences testables et vérifiables (Zemín et al., 2025), ainsi que **iii) l'utilisation de métriques standards** telles que QUS et INVEST afin de permettre l'évaluation et la comparaison quantitative des modèles.

GeneUS (Rahman et al., 2024) ou UStAI (Yamani et al., 2025) n'explorent qu'un sous-ensemble de ces attributs ce qui rend la comparaison quantitative difficile. Par conséquent, nous concentrons notre évaluation sur la démonstration de l'efficacité de `T2S` par ablation ainsi que son évaluation sur des métriques standards plutôt qu'en comparatif direct de l'état de l'art. Le tableau 1 répertorie les différences entre les approches antérieures et la nôtre.

TAB. 1 – *Comparatif des Fonctionnalités des Modèles de Génération d'US.*
***Légende : AC** (Critères d'Acceptation), **Métriques Std.** (Métriques Standards : INVEST/QUS).*

Fonctionnalité	**GeneUS** (Rahman et al., 2024)	**UStAI** (Yamani et al., 2025)	**`T2S`** (La Nôtre)
Génération Contextualisée (US)	×	×	✓
Génération d'**AC**	✓	×	✓
Évaluation via **Métriques Std.**	×	✓	✓

TAB. 2 – *Évaluation de Text2Stories & des Ablations*
***Gras = `T2S` est 1er**, Souligné = `T2S` est 2ème*

Méthode	**Alignement Humain-Modèle**				**Défauts de Format et d'Unicité**				
	INVEST (Wake, 2003)				**AQUSA Score (QUS)** (Lucassen et al., 2016)				
	Acc ↑	Prec ↑	Recall ↑	F1 ↑	Atom. ↓	Well-Fo. ↓	Mini. ↓	Unif. ↓	Uniq. ↓
LLM	0.93	0.93	1.00	0.96	0.00	0.00	0.00	0.00	0.54
LLM + CoT	0.73	0.75	0.93	0.82	0.20	0.00	0.04	0.12	0.00
RAG + CoT	0.77	0.82	0.89	0.85	0.16	0.00	0.00	0.12	0.00
`T2S` (Gen)	0.83	0.85	0.96	0.90	0.08	**0.00**	**0.00**	0.42	0.12
`T2S` (Deg)	0.51	0.74	0.49	0.58	0.44	0.00	0.16	0.24	0.00

4.3 Résultats Empiriques

Nous avons étudié les résultats de `T2S` via deux ensembles de métriques standards en ingénierie des exigences : **i) INVEST** capture des concepts haut niveau relevant de la pertinence industrielle et l'applicabilité des US (Wake, 2003), **ii) QUS**, et plus précisément le modèle AQUSA (Lucassen et al., 2015, 2016), se concentre principalement sur la validité syntaxique et le respects des règles de gestion applicables aux US.

Une analyse de la table 2 met en évidence plusieurs tendances. En termes d'accord humain-modèle (INVEST), `T2S` surpasse systématiquement `LLM+CoT` et `RAG+CoT`. Le `LLM` surpasse tous les autres modèles, ce qui est contre-intuitif dans la mesure où les *Chain-of-Thought* sont meilleures pour de telles tâches (Wei et al., 2022). Les scores AQUSA montrent que le LLM est le meilleur, sauf en matière d'unicité, où il se classe dernier, tandis que `T2S` arrive en deuxième position. Il s'agit là d'une observation cruciale, car elle signifie que le `LLM` a produit 14 US non uniques sur 25 éléments, soit près de la moitié des US générées.

Enfin, `T2S` affiche des performances inférieures en matière d'uniformité en raison d'un mélange de langues dans la sortie (anglais et français). En effet, le modèle RAG dans le bloc d'élicitation (section 2) est exposé à des données en français et en anglais. La traduction a posteriori entraîne des différences de formulation, d'où un score d'uniformité inférieur.

5 Discussion & Conclusion

En conclusion, nous avons présenté Text2Stories (T2S), un outil multi-agents permettant la génération et l'évaluation des *user stories* (US). Notre modèle se distingue de l'état de l'art en permettant une génération contextualisée d'*user stories* avec des *epic stories*, des critères d'acceptation, ainsi qu'une évaluation INVEST automatique effectuée par le modèle, et dont une partie a été validée par 20 experts humains.

T2S démontre un meilleur alignement avec le jugement humain que les ablations LLM+CoT et RAG+CoT concernant les principes INVEST. T2S est également compétitif sur QUS en se classant premier en termes de minimalité et de forme (*Well-Formed*), et deuxième en termes d'atomicité et d'unicité. À l'inverse, le LLM, qui se classe en tête dans la plupart des catégories, affiche les pires performances en matière d'unicité. En d'autres termes, il génère des récits d'utilisateurs correctement formatés mais hautement redondants, contrairement aux modèles de raisonnement (LLM+CoT, RAG+CoT et T2S).

Par conséquent, notre agent offre des performances globales compétitives en se classant deuxième en matière d'alignement humain sur INVEST et premier ou deuxième pour QUS, à l'exception de l'uniformité. De plus, T2S obtient une meilleure unicité que le LLM, ce qui est souhaitable pour une génération diversifiée de *user stories*. En d'autres termes, T2S apparaît plus adapté pour des générations longues, tandis que le LLM se prête mieux à des générations courtes, pour lesquelles la redondance n'est pas à prendre en compte.

Références

Chae, Y. et T. Davidson (2025). Large Language Models for Text Classification : From Zero-Shot Learning to Instruction-Tuning. *Sociol. Methods Res.*, 00491241251325243. Publisher : SAGE Publications Inc.

Dalpiaz, F., I. van der Schalk, S. Brinkkemper, F. B. Aydemir, et G. Lucassen (2019). Detecting terminological ambiguity in user stories : Tool and experimentation. *Information and Software Technology 110*, 3–16, doi: 10.1016/j.infsof.2018.12.007.

Lewis, P., E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, et D. Kiela (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *NeurIPS*, Volume 33, pp. 9459–9474. Curran.

Lucassen, G., F. Dalpiaz, J. M. E. M. van der Werf, et S. Brinkkemper (2016). Improving agile requirements : the Quality User Story framework and tool. *RE 21*(3), 383–403, doi: 10.1007/s00766-016-0250-x.

Lucassen, G., F. Dalpiaz, J. M. v. d. Werf, et S. Brinkkemper (2015). Forging High-Quality User Stories : Towards a Discipline for Agile Requirements. *RE 2015*, 126–135, doi: 10.1109/RE.2015.7320415. Publisher : IEEE.

Neo, G. d. S., J. Moura, H. Almeida, A. Neo, et O. F. Júnior (2024). User Story Tutor (UST) to Support Agile Software Developers. In *CSEDU*, Volume 2, pp. 51–62. SCITEPRESS, doi: 10.5220/0012619200003693.

R. Amna, A. et G. Poels (2021). Systematic Literature Mapping of User Story Research. *IEEE Access*, 51723–51746.

Raharjana, I. K., D. Siahaan, et C. Fatichah (2021). User Stories and Natural Language Processing : A Systematic Literature Review. *IEEE Access 9*, 53811–53826, doi: 10.1109/ACCESS.2021.3070606.

Rahman, T., Y. Zhu, L. Maha, C. Roy, B. Roy, et K. Schneider (2024). Take Loads Off Your Developers : Automated User Story Generation using Large Language Model. In *IEEE ICSME 2024*, pp. 791–801, doi: 10.1109/ICSME58944.2024.00082. ISSN : 2576-3148.

Ronanki, K., B. Cabrero-Daniel, et C. Berger (2024). ChatGPT as a Tool for User Story Quality Evaluation : Trustworthy Out of the Box ? In P. Kruchten et P. Gregory (Eds.), *XP 2022-23*, Cham, pp. 173–181. Springer Nature Switzerland, doi: 10.1007/978-3-031-48550-3$_1$7.

Schluntz, E. et B. Zhang (2024). Building Effective AI Agents.

Tawosi, V., A. Al-Subaihin, R. Moussa, et F. Sarro (2022). A versatile dataset of agile open source software projects. In *MSR 2022*, New York, NY, USA, pp. 707–711. Association for Computing Machinery, doi: 10.1145/3524842.3528029.

Wake, B. (2003). INVEST in Good Stories, and SMART Tasks - XP123.

Wei, J., X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, et D. Zhou (2022). Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, Red Hook, NY, USA, pp. 24824–24837. Curran Associates Inc.

Yamani, A., M. Baslyman, et M. Ahmed (2025). Leveraging LLMs for User Stories in AI Systems : UStAI Dataset. In *PROMISE 2025*, New York, NY, USA, pp. 21–30. Association for Computing Machinery, doi: 10.1145/3727582.3728689.

Yao, S., J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, et Y. Cao (2023). ReAct : Synergizing Reasoning and Acting in Language Models. *ICLR 2023*.

Zemín, L., A. Godio, C. Cornejo, R. Degiovanni, S. Gutiérrez Brida, G. Regis, N. Aguirre, et M. F. Frias (2025). An Empirical Study on the Suitability of Test-based Patch Acceptance Criteria. *TOSEM 34*(3), 1–20, doi: 10.1145/3702971. Publisher : ACM.

Summary

User stories (US) are a key artifact in software engineering inherited from Agile methods that allow functional and non-functional requirements to be expressed from the end user's perspective. However, their study in natural language processing has several limitations. Agile and open source methodologies are partially incompatible: the former assumes close, colocated collaboration, while the latter relies on distributed, asynchronous teams. As a result, there exists few public datasets, and none that combine high-quality US, established metrics, and contextual project elements. In practice, US are often poorly formulated and evaluated through informal and oral reviews such as the "Three Amigos" ceremony. These limitations hinder the progress in US's automatic generation and evaluation. This article presents an Multi-Agents AI tool that generates and evaluates contextualized US, providing both comparable and reproducible results for researchers and practitioners.