Automated Unobtrusive Techniques for Linking requirements and testing in agile software development AUTOLINK PROJECT

Fernando Pastor Ricós[1], Ashley van Can[2], Beatriz Marín[1], Fabiano Dalpiaz[2], Tanja Vos[1,3]

[1]Universitat Politècnica de València (UPV), Spain

[2]Utrecht University (UU), The Netherlands

[3]Open Universiteit (OU), The Netherlands

Abstract. *Software professionals often face challenges in defining requirements and conducting thorough testing, particularly within agile development environments. The rapid and frequent iterations of agile methods encourage developers to take shortcuts to meet tight release deadlines, which may lead to low-quality software. To enhance software quality, the AUTOLINK project explores the synergies between requirements engineering and software testing, aiming to seamlessly integrate them into agile environments. AUTOLINK seeks to develop techniques that enable practitioners to improve software quality by testing it against its requirements in a way that is unobtrusive, cost-effective, and requires minimal effort.*

1. Introduction

Many of the recent disasters that occurred due to software faults are rooted in the human aspects of software engineering: "developers don't understand the problem they're trying to solve" [Somers 2017]. As a consequence, "the serious problems that have happened with software have to do with requirements, not coding errors [Somers 2017]. Hence, representing requirements and testing the developed software against them is crucial.

Over the past decades, the release of the agile manifesto [Beck et al. 2001] led to unprecedented changes in the methods that software development teams employ to build software. Traditional approaches borrowed from systems engineering (e.g., waterfall, the V-model) have been gradually replaced with methods that focus on the quick release of working software (e.g., Scrum, Kanban, XP). The World Quality Report [SOGETI 2019] has been arguing for years that this has led to blurred boundaries between development and quality assurance techniques like requirements engineering and testing. Consequently, serious attention needs to be given to the study of intelligent automation of end-to-end testing based on requirements, to truly provide business assurance, as organizations scale to enterprise agility [SOGETI 2019].

Software engineering research has studied agile development in many ways [Dingsøyr et al. 2012], especially in terms of human collaboration [Conboy et al. 2010] and its relationships with innovation theories [Nerur and Balijepally 2007]. Despite the acknowledged impact of agile development on requirements engineering and testing [Paetsch et al. 2003], there is a substantial lack of studies that describe how requirements are specified, managed, and tested in agile development [Inayat et al. 2015]

[Schön et al. 2017]. This is a serious issue for robust approaches that focus on the problem to solve and on validating its solution, like the V-model, were replaced by alternatives that aim at development and release speed [Bosch 2015], at the cost of decreasing software reliability [da Costa et al. 2016].

Concretely, user stories are the primary tool for managing requirements in agile development, with adoption rates reaching 90% [Cohn 2004, Kassab 2015, Lucassen et al. 2016]. However, they are often created hastily without adhering to quality standards. Although poorly written user stories may spark team discussions [Lucassen et al. 2017], they are difficult to trace back to test cases and code. Regarding testing, most testing frameworks, like Gherkin and Cucumber, rely on scripts to automate manual steps. These tools support BDD by generating test scripts from user stories and automating acceptance criteria checks. However, their lack of flexibility to adapt to changes results in high maintenance efforts [Vos and Aho 2017, Binamungu et al. 2018].

The AUTOLINK project aims to tackle these challenges by developing innovative tools, powered by natural language processing and machine learning, that integrate requirements engineering with software testing in agile practices. By leveraging unobtrusive software tools, this approach seeks to encourage companies to adopt these techniques more rigorously, ultimately resulting in higher-quality software.

2. Goals of the AUTOLINK Project

The main goals of the AUTOLINK Project are:

- *O1. Enhancing traceability of requirements*: To support practitioners in validating their system functionalities, we develop techniques to automatically extract testing-relevant behavioral information from requirements, which can serve as a starting point for testing activities (including manual, automated, and scriptless testing). In addition, achieving ubiquitous traceability involves ensuring that trace links between software artifacts (e.g., from requirements to code) are consistently available without requiring extra effort. By promoting the disciplined creation of user stories, we will apply artificial intelligence techniques to extract trace links between requirements and the software.
- *O2. Automating requirements validation through scriptless testing*: Trace links support requirements validation by connecting user stories to software artifacts. To provide value while avoiding the maintenance challenges of test scripts, we will explore how user stories, enriched with acceptance criteria, can be automatically validated using scriptless testing. A key challenge lies in guiding intelligent agents to select the appropriate actions and determine the next tests to perform.
- *O3. Designing seamless tools to enhance agile development*: We will develop a tool-supported approach that incorporates the results of O1 and O2, designed for integration into software companies. Our goal is to ensure the method is seamless and unobtrusive, allowing it to blend into existing development practices without necessitating significant changes to developers' workflows.
- *O4. Collecting evidence of value for software companies through empirical studies*: The tools and methods developed in O3 will allow us to assess the practical impact of the automated techniques from O1 and O2. By conducting case studies with partnering software companies we aim to gather empirical evidence demonstrating the benefits our approach offers to software developers.

3. Beneficiaries

The AUTOLINK project is a collaborative research project emerges from research groups that focus on different, yet complementary sub-fields of software engineering: automated scriptless testing and requirements engineering. These groups are: Open Universiteit (OU), The Netherlands; Utrecht University (UU), The Netherlands, and Universitat Politècnica de València (UPV), Spain. Also, six industry partners: ING Bank, Mendix Technology, TestCompass, Newspark, AXINI, and B00, are actively involved. Collaboration with these industry partners is essential for testing solutions and addressing challenges in real-world environments, ensuring the project's practical impact and relevance.

4. Results

The results of the project so far are:

- In collaboration with Mendix, we analyzed real-world backlog items, aiming to determine how requirements are documented in agile teams, and where additional information relevant to the project is stored.
- A collection of open-source issue tracking systems (ITSs) of real projects, in addition to the studied ITSs of Mendix, was analyzed to determine how requirements are represented in practice. In this study, we found that ITSs lack an organized and standardized representation of requirements. The identified challenges led to the experimentation with LLMs on the automated identification and classification of requirements [van Can and Dalpiaz 2025].
- A study to compare AI-based techniques as well as their performance against humans to derive domain models from collections of user stories was performed.
- A state model is inferred while testing at the GUI level with the scriptless testing tool TESTAR. To do that, an action selection mechanism has been designed and implemented to derive all possible actions from the available widgets.
- A listener mode is implemented such that TESTAR state models can also be inferred when the software is being tested by other means than TESTAR (i.e., manual testing, scripted automation, etc.).
- A preliminary prototype for improved action selection incorporating the use of LLMs into the TESTAR loop that has been tried out on open-source web applications, including prompts containing user requirements. This prototype is available at `https://github.com/TESTARtool/TESTAR_dev/tree/autolink` with an open-source 3-Clause BSD License.
- An industrial case study was performed using the LLM-empowered TESTAR at ING on real websites including chatbot functionality for requirements.

Results of AUTOLINK show that the project is aligned with the latest developments in GenAI and Large Language Model (LLM) technologies, which strongly support the research goals of the project. While our initial efforts concentrated on improving action selection mechanisms through reinforcement learning, the incorporation of state-of-the-art LLMs has broadened our scope. This integration holds significant promise for achieving our ambitious initial objectives: creating advanced tools capable of intelligently and autonomously extracting, connecting, and validating user requirements.

Acknowledgement This research is partially funded by the Dutch Research Council (NWO) through the Open Technology Programme 2021-II TTW, project AUTOLINK (19521), 2023-2027.

References

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al. (2001). The agile manifesto.

Binamungu, L. P., Embury, S. M., and Konstantinou, N. (2018). Maintaining behaviour driven development specifications: Challenges and opportunities. In *25th SANER*, pages 175–184. IEEE.

Bosch, J. (2015). Speed, data, and ecosystems: the future of software engineering. *IEEE Software*, 33(1):82–88.

Cohn, M. (2004). *User stories applied: For agile software development*. Addison-Wesley Professional.

Conboy, K., Coyle, S., and Wang, X. (2010). People over process: Key challenges in agile development,(99), 48–57.

da Costa, D., McIntosh, S., Kulesza, U., and Hassan, A. (2016). The impact of switching to a rapid release cycle on the integration delay of addressed issues: an empirical study of the mozilla firefox project. In *13th MRS*, pages 374–385.

Dingsøyr, T., Nerur, S., Balijepally, V., and Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development.

Inayat, I., Salim, S. S., Marczak, S., Daneva, M., and Shamshirband, S. (2015). A systematic literature review on agile requirements engineering practices and challenges. *Computers in human behavior*, 51:915–929.

Kassab, M. (2015). The changing landscape of requirements engineering practices over the past decade. In *5th EmpiRE Workshop*, pages 1–8. IEEE.

Lucassen, G., Dalpiaz, F., van der Werf, J., and Brinkkemper, S. (2016). The use and effectiveness of user stories in practice. In *22nd REFSQ*, pages 205–222. Springer.

Lucassen, G., Dalpiaz, F., van der Werf, J., and Brinkkemper, S. (2017). Improving user story practice with the grimm method: A multiple case study in the software industry. In *23rd REFSQ*, pages 235–252. Springer.

Nerur, S. and Balijepally, V. (2007). Theoretical reflections on agile development methodologies. *Communications of the ACM*, 50(3):79–83.

Paetsch, F., Eberlein, A., and Maurer, F. (2003). Requirements engineering and agile software development. In *12th WET ICE*, pages 308–313. IEEE.

Schön, E.-M., Thomaschewski, J., and Escalona, M. J. (2017). Agile requirements engineering: A systematic literature review. *Computer standards & interfaces*, 49:79–91.

SOGETI (2019). The world quality report 2019-2020.

Somers, J. (2017). The coming software apocalypse. *The Atlantic*, 26:1.

van Can, A. T. and Dalpiaz, F. (2025). Locating requirements in backlog items: Content analysis and experiments with large language models. *Information and Software Technology*, 179:107644.

Vos, T. and Aho, P. (2017). Searching for the best test. In *IEEE/ACM 10th International Workshop on Search-Based Software Testing (SBST)*, pages 3–4. IEEE.