Where Requirements and Agility Meet: No Man's Land or a Land of Opportunity?

Fabiano Dalpiaz and Jan-Philipp Steghöfer

Is managing requirements in agile development different from managing them in non-agile settings? In practical settings, are there differences in the techniques that are most effective for the elicitation, prioritization, specification, and validation of these requirements?

According to the International Requirements Engineering Board (IREB), there are certainly some peculiarities, as they have released an advanced level certification (Aschauer, 2019) that defines RE@Agile as a cooperative, iterative and incremental approach where the traditional requirements engineering (RE) activities of eliciting, agreeing upon, and documenting requirements are conducted in alignment with the principles of the Agile Manifesto.

Similarly, practitioners have proposed numerous frameworks and innovations, such as user stories (Cohn, 2004), the SAFe scaled agile framework (Leffingwell, 2019), techniques for writing testable requirements like Gherkin/Cucumber, user story mapping, the INVEST framework for writing better user stories, definitions of ready and done, and more.

But has the research in requirements and software engineering evolved accordingly? Are academics publishing theoretical, technological, and empirical studies that take the nature of RE in agile development into account? While some efforts exist, not enough research is devoted to the important intersection between requirements engineering and agile development, leading to a disconnect between research and practice.

The following pages are a call for action for the research community to study RE in agile contexts in more depth. We base our reflections on our mixed background: on the one hand, we have been conducting research on RE in agile contexts for several years; on the other hand, we have been involved in industrial projects and collaborated with software development companies through several R&D projects.

A jungle of industry-pushed innovations and gurus

When it comes to agile development, most theories and innovations do not originate and are not pushed top-down from the "ivory tower" of academia. Rather, they come from practice. The influential Agile Manifesto has been written and signed by seventeen professionals who identified similarities in approaches like Scrum, eXtreme Programming, and the like. User stories as we know them emerged from the individual work of Ron Jeffries and then Mike Cohn.

The INVEST framework is a mnemonic (indicating that good user stories should be Independent, Negotiable, Valuable, Estimable, Small, and Testable) that was coined by Bill Wake. None of these people are academics.

These innovations have been widely adopted by fellow practitioners, who found pragmatic, no-nonsense solutions to their real problems. Whether these solutions had a solid and proven theoretical underpinning – the holy grail of academia – was a marginal aspect. Some of these people have embraced these innovations to the point of becoming trainers, book writers, agile mentors, sometimes even gurus of the wild world of agile development.

While this observation applies to agile development in general, not only to RE and agile, what we find remarkable is that only few researchers have decided to pick RE in agile contexts as their primary research topic. A systematic mapping study of RE in agile software development (Curcio, 2018) shows a total of 104 studies published between 2001 and 2017; this is not a long list to begin with – but what is more, the authors identify two challenges: quality requirements are often neglected, and empirical evaluation studies are largely missing.

The latter gap shows a clearly missed opportunity, as researchers could provide the scientific underpinning regarding the effectiveness of the industry-pushed innovations. This is one of the reasons that motivated us to propose and organize the first AgileRE workshop at the REFSQ 2024 conference, which we see as a step toward fostering a community to deliver scientific studies around these topics.

Are user stories requirements?

The seminal book by Mike Cohn (Cohn, 2004) puts forward user stories as a (the) key component of the product backlog, as they define user-oriented requirements by specifying *who* needs a certain functionality, *what* is requested, and *why* implementing this feature would be beneficial.

User stories are a leading artifact that represents requirements in agile contexts, but not the only one. Epics and themes are common ways to organize collections of user stories that pertain to the same topic. Acceptance criteria specify precise conditions under which a user story is met. Moreover, techniques such as feature and example mapping (Berends, 2021) describe how a user story can be refined.

Nevertheless, in academic contexts, we often hear statements such as "but user stories are not requirements...". Perhaps such claims arise because user stories are not meant to be used as a contractual specification of the system to-be, but they rather aim to foster the conversation about requirements within the team: *customer collaboration over contract negotiation* is one of the principles of the Agile Manifesto. Yet, the expectation that there would be a software specification document with hundreds of "shall" statements does not *generally* hold true in agile contexts, although this still sometimes happens in more rigid, established organizations.

Therefore, user stories shall be treated, together with their related artifacts and techniques, as a representation of requirements.

No requirements: code is king!

At the same time, even the notion of user stories in a backlog is being challenged. Developers increasingly see requirements as just another artifact that should ideally be managed and maintained in a version control system. Everything-as-code means that requirements are captured in a textual format that lends itself to merging and diffing with the same tools developers use for the actual programs they write and is fully integrated into the developer workflow inside an IDE. Approaches such as behavior-driven development (Wynne, 2017) and languages like Gherkin[1] make this simple and add the benefit that the requirement specification is also an executable test specification. Other tools such as TReqs (Knauss et al., 2018) combine requirements management with traceability. While TReqs is an example of a tool driven by academics, its origin is in the needs of an industrial partner and their desire to manage requirements pragmatically and non-intrusively.

In all of these cases, developers do not need to deal with an additional tool. All information, including the requirements, is stored in the codebase. If the development teams sharing this information employ proper code hygiene, all information is consistent across the codebase in every commit – requirements, code and tests are always in sync. The disadvantage is that it is more difficult to provide access to the requirements to external stakeholders, e.g., a customer representative. In situations where project managers are the interface provided by customers, there is a preference for more approachable web-based tools such as JIRA. Nonetheless, requirements-as-code is a reality even in industrial settings, but has not yet received a lot of attention from the research community.

Managing requirements, products, or projects?

Many people who deal with requirements do not self-identify as requirements engineers. Herrmann (2013) had already identified this task-job title discrepancy over a decade ago. In the second author's organization, there is no explicit "requirements engineer" role and the tasks usually attributed to that role are performed by a product owner or usability engineering experts. The former are mostly responsible for keeping requirements up-to-date (requirements management) while the latter are involved in elicitation and provide methodological support. Developers, on the other hand, support the requirements management activities by providing continuous feedback.

This division of labor may not be surprising, as the organization uses Scrum, an agile framework that does not include an explicit RE role. But in many of the customer organizations, even those with high maturity, requirements engineers are few and far between. Instead, the boundaries between project management and requirements engineering are blurred. Tools like JIRA

[1] https://cucumber.io/docs/gherkin/

support this blurring as they are used as the main tool to manage project progress and planning based on requirements in the form of epics and user stories, thereby merging requirements representation and project management (van Can, 2024). Work breakdown structures replace detailed Gantt charts, and detailed long-term planning is no longer a focus.

This is also visible in the contracts between suppliers and customers. They are increasingly not bound to the delivery of a set of features captured in a product requirements document. Instead, they are based on "time and material", i.e., customers commit to a certain budget for development time and other expenses. During that time, the suppliers and customers co-develop the product vision and react to changing circumstances. The traditional product management approaches are therefore becoming obsolete even in highly regulated industries such as medical devices and automotive.

Conclusion

We have heard from some of our esteemed colleagues that research in agile RE is a relic of the last decade. We wholeheartedly disagree – in contrast, we believe that agile RE is more relevant than ever. While researchers still debate whether user stories are requirements, practitioners are using pragmatic solutions that "just work" and shift the way of working in organizations. The benefits that scientific inquiry brings to these solutions – systematic and unbiased evaluations, theoretical grounding, reproducibility, etc. – are not currently realized as the body of work does not consider agile RE holistically.

Therefore, we issue a call to action: for academics to stop bickering about terminological details and, instead, setting your eyes on the realities on the ground and support the industrial practices with your methodological prowess and your scientific insights; for industry to provide access to academics and benefit from the theoretical and empirical results that research teams can provide.

Agile RE is the reality in industry – let's work together to align theory with practice and truly realize its potential. We are more than happy to hear from you.

References

Aschauer, B., Hruschka, P., Lauenroth, K., Meuten, M., & Rogers, G. (2019). Handbook of RE@Agile According to the IREB Standard. *International Requirements Engineering Board*.

Berends, J., & Dalpiaz, F. (2021). Refining user stories via example mapping: an empirical investigation. In *2021 IEEE 29th International Requirements Engineering Conference (RE)* (pp. 345-355). IEEE.

Van Can, A., & Dalpiaz, F. (2024). Requirements Information in Product Backlog Items: Content Analysis. In Requirements Engineering: Foundation for Software Quality. REFSQ 2024, Springer.

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

Curcio, K., Navarro, T., Malucelli, A., & Reinehr, S. (2018). Requirements engineering: A systematic mapping study in agile software development. *Journal of Systems and Software*, *139*, 32-50.

Leffingwell, D. (2016). *SAFe® 4.0 reference guide: scaled agile framework® for lean software and systems engineering*. Addison-Wesley Professional.

Herrmann, A. (2013). *Requirements Engineering in Practice: There Is No Requirements Engineer Position.* In: Doerr, J., Opdahl, A.L. (eds) Requirements Engineering: Foundation for Software Quality. REFSQ 2013. Lecture Notes in Computer Science, vol 7830. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-37422-7_25

Knauss, Eric, Grischa Liebel, Jennifer Horkoff, Rebekka Wohlrab, Rashidah Kasauli, Filip Lange, and Pierre Gildert. "T-reqs: Tool support for managing requirements in large-scale agile system development." In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pp. 502-503. IEEE, 2018.

Wynne, M., Hellesoy, A., & Tooke, S. (2017). *The Cucumber book: Behaviour-driven Development for Testers and Developers*. Pragmatic Bookshelf.