Improving Clarity and Completeness in User Stories: Insights from a Multi-Domain Analysis with Developer Feedback

Maria Regina Araújo Souza[a] and Tayana Conte[b]
Federal University of Amazonas, Manaus, Brazil
{*maria.souza, tayana@icomp.ufam.edu.br*

Keywords: Software Requirements, User Stories, Agile Methodologies, Requirements Analysis.

Abstract: The clarity and completeness of requirements are crucial in agile software development, where user stories are widely used to capture user needs. However, poorly written user stories can introduce ambiguities, leading to inefficiencies in the development process. This paper presents a detailed analysis of 30 user stories from five different domains, along with feedback from 50 developers gathered through a questionnaire. The analysis, based on the INVEST criteria (Independent, Negotiable, Valuable, Estimable, Small, and Testable), identified common issues such as vague acceptance criteria, insufficient technical details, and overly broad stories. Based on these findings, we present targeted recommendations for improving user story quality, including refining acceptance criteria, breaking down large stories into smaller components, and incorporating adequate technical details. The feedback from developers reinforced the value of these practices, highlighting the importance of collaboration in refining user stories. This study offers actionable insights and practical strategies to enhance user story quality and promote continuous improvement in agile software development.

1 INTRODUCTION

In agile software development, user stories are widely used to capture requirements concisely and facilitate communication between development teams, stakeholders, and clients (Cohn, 2004; Leffingwell, 2010). They help align software features with business objectives and user needs (Williams and Cockburn, 2003). However, despite their advantages, many user stories lack clarity, well-defined acceptance criteria, and technical details, leading to inefficiencies in agile projects (Inayat et al., 2015; Lucassen et al., 2016).

Ambiguous user stories contribute to miscommunication, rework, and project delays (Lucassen et al., 2015; Heck and Zaidman, 2018). Poorly structured stories hinder development efficiency and compromise stakeholder satisfaction (Leffingwell, 2018). Given their importance in agile methodologies, improving the clarity and completeness of user stories is essential to reducing development challenges and enhancing project outcomes.

This paper presents an in-depth analysis of 30 user stories from five distinct domains—E-commerce, Billing, Security, Collaboration Tools, and E-learning. Using the INVEST criteria (Independent, Negotiable, Valuable, Estimable, Small, and Testable), we identify common deficiencies and propose actionable improvements. These include refining acceptance criteria, ensuring sufficient technical details, and breaking down complex stories into smaller, more manageable tasks.

Additionally, to validate these findings, we conducted a developer survey with 50 participants, gathering insights into real-world challenges when working with user stories. The objective of this research is to provide practical recommendations that support developers in writing clearer, more detailed, and actionable user stories, contributing to improved agile development practices.

[a] https://orcid.org/0000-0001-9705-3894
[b] https://orcid.org/0000-0001-6436-3773

2 RELATED WORK

Several studies have analyzed the quality of user stories and their impact on agile development. Wake (2003) introduced the INVEST criteria, which serve as a guideline for writing effective user stories by ensuring they are actionable, valuable, and testable. Despite these guidelines, research indicates that user stories frequently fail to meet these standards in practice.

Souza, M. R. A. and Conte, T.
Improving Clarity and Completeness in User Stories: Insights from a Multi-Domain Analysis with Developer Feedback.
DOI: 10.5220/0013363500003929
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 27th International Conference on Enterprise Information Systems (ICEIS 2025) - Volume 2*, pages 272-279
ISBN: 978-989-758-749-8; ISSN: 2184-4992
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

Lucassen et al. (2016, 2015) identify recurring issues such as vague acceptance criteria, lack of technical details, and oversized stories. These deficiencies result in ambiguities and inefficiencies during development. To address these challenges, Lucassen et al. (2016) proposed the Quality User Story Framework (QUS), which aims to improve clarity and consistency. Empirical studies suggest that applying QUS enhances alignment between stakeholder expectations and development outcomes.

Inayat et al. (2015) conducted a systematic review of agile requirements engineering practices and found that unclear user stories and insufficient details frequently lead to project delays and misaligned expectations. Similarly, Heck and Zaidman (2018) highlight the importance of testability and consistency in reducing ambiguities and improving team collaboration.

From a broader perspective, Leffingwell (2010) discusses the role of user stories in large-scale agile projects, advocating for structured approaches such as the Scaled Agile Framework (SAFe). Meanwhile, Williams and Cockburn (2003) emphasize the importance of continuous feedback in refining user stories to align with evolving user needs.

This study builds upon these works by not only identifying recurring issues in user stories but also validating them through direct developer feedback. Our approach provides actionable recommendations based on real-world challenges, offering practical insights to enhance user story quality in agile software development.

3 RESEARCH METHODOLOGY

This study employed a mixed-methods approach combining qualitative and quantitative methods to evaluate user story quality in agile development. The research was conducted in three phases:

Data Collection: User stories were gathered from publicly available repositories across five domains: E-commerce, Billing, Security, Collaboration Tools, and E-learning. The selection ensured diversity in complexity, application contexts, and functional scope.

User Story Evaluation: Stories were assessed using the INVEST criteria (Independent, Negotiable, Valuable, Estimable, Small, and Testable), identifying common issues such as unclear acceptance criteria, lack of technical details, and overly broad scopes.

Developer Feedback Validation: A developer questionnaire was designed to collect insights on real-world challenges related to user stories, including clarity, technical specifications, and acceptance criteria. The questionnaire was shared in private corporate networks of companies focused on software development—particularly web development—, on LinkedIn, and was also distributed through professional connections, leveraging a network of developers actively working in agile environments. The feedback gathered helped refine the recommendations for improving user story quality.

Figure 1 illustrates this methodology, emphasizing the interconnection of these phases and the balance between theoretical analysis and practical developer insights.

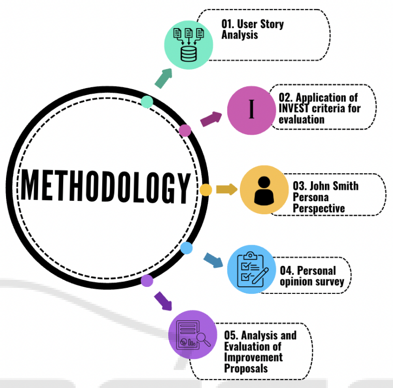
METHODOLOGY
01. User Story Analysis
I
02. Application of INVEST criteria for evaluation
03. John Smith Persona Perspective
04. Personal opinion survey
05. Analysis and Evaluation of Improvement Proposals

Figure 1: Methodology Overview.

3.1 User Story Analysis

We conducted an extensive analysis of 30 user stories from publicly available repositories, covering various domains such as e-commerce, billing, security, collaboration tools, and e-learning. The complete dataset source is available in an external document: Supporting Documentation for User Story Analysis.

Each user story was evaluated not only using the INVEST criteria, but also through a more developer-centric perspective. While the INVEST framework provides a structured guideline for user story quality, our analysis revealed that even stories that adhered to these principles still lacked key elements from a developer's standpoint.

To capture these real-world challenges, we incorporated the perspective of John Smith, a full-stack developer persona. This approach highlighted additional critical aspects often overlooked in standard user story evaluations, such as missing technical details, unclear problem descriptions, and insufficient actionable value. While these stories were generally aligned with INVEST principles, the developer's perspective underscored gaps that could lead to inefficiencies, misunderstandings, and additional implementation effort.

3.2 Developer Feedback Questionnaire

To validate the findings from the user story analysis, we designed a questionnaire targeting 50 developers across various domains. The goal was to identify real-world challenges in user story usage and gather insights for improvement.

The questionnaire was divided into multiple sections, each focusing on key aspects of user story quality:

Table 1: Key Aspects of User Story Quality Assessed.

Aspect	Description
Clarity	Assesses whether user stories avoid vague descriptions or ambiguous goals.
Acceptance Criteria	Evaluates if acceptance criteria are explicitly defined and sufficient for validation.
Technical Details	Checks whether stories provide necessary technical details, such as API references and data formats.
Improvements	Gathers developer feedback on enhancing clarity, technical completeness, and usability.

The questionnaire was distributed via private company networks, LinkedIn agile communities, and referrals from professionals in software development. The complete questionnaire is available at: Supporting Documentation for User Story Analysis.

4 RESULTS AND DISCUSSION

4.1 User Story Analysis

To ensure a comprehensive analysis, the 30 collected user stories were organized into five domains: E-commerce, Billing, Security, Collaboration Tools, and E-learning. This categorization facilitated the identification of domain-specific challenges, particularly those related to clarity, technical details, and acceptance criteria.

The primary analysis was conducted from the perspective of the persona **John Smith**, a full-stack developer with five years of experience. This approach provided practical insights into how developers interpret and implement user stories, revealing real-world challenges such as ambiguous descriptions, missing technical details, and unclear acceptance criteria.

To complement this analysis, the INVEST criteria were used as a secondary framework to systematically assess user story structure and completeness. While many stories aligned with INVEST principles, gaps remained in technical details and clarity from a developer's standpoint. The full analysis, including detailed evaluations and persona documentation, is available at: Supporting Documentation for User Story Analysis.

A summary of the key findings from the user story analysis is presented in Table 2, highlighting the most relevant challenges identified within each domain.

Table 2: Summary of User Stories Analysis Across Domains.

User Story and Domain	Key Observations
E-commerce: User Story 1.1 - "As a visitor, I want to view a selection of shirts to explore my options."	Lacked clear filtering criteria and UI specifications. Needed structured acceptance criteria.
Billing: User Story 2.1 - "As a user, I want to create, update, and delete a subscription."	Combined multiple actions, making estimation difficult. Required breakdown into smaller tasks.
Security: User Story 3.1 - "As a security engineer, I want to verify the system configuration to ensure secure settings."	Lacked security benchmarks and validation tools. Needed supplementary documentation.
E-learning: User Story 4.1 - "As a student, I want to practice grammar exercises to improve my language skills."	Lacked content format specification (multiple-choice vs. fill-in-the-blank). Required clearer acceptance criteria.

4.1.1 Domain-Specific Challenges and Observations

Across the five domains analyzed, common challenges were identified, making it difficult for developers like John Smith to implement them effectively. Table 3 summarizes the key findings and recommendations from the analysis.

Table 3: Domain-Specific Challenges and Suggested Improvements.

Identified Challenges	Recommended Improvements
Lack of Clear Acceptance Criteria	Define explicit, testable criteria to ensure alignment with development expectations.
Overly Complex User Stories	Break down large stories into smaller, independent tasks for better estimability and clarity.
Insufficient Technical Details	Include essential specifications such as API references, security requirements, and UI/UX guidelines.
Broad and Ambiguous Functionalities	Ensure stories focus on a single functionality to prevent excessive scope and implementation issues.

4.2 Discussion of Key Findings

From the detailed analysis of the user stories across these domains, several recurring patterns were observed:

1. **Clear Acceptance Criteria:** Many stories lacked well-defined acceptance criteria, making it difficult for developers to determine when the story was complete. This issue was particularly evident in *User Story 2.1*, where no explicit validation criteria were provided for subscription management actions. Providing detailed acceptance criteria would reduce ambiguity and streamline both development and testing processes.
2. **Breaking Down Complex Stories:** A significant number of stories were overly complex, attempting to cover multiple functionalities within a single story. For example, in the Billing domain, a single story combined actions such as creating, updating, and deleting subscriptions, making estimation difficult. Breaking these stories into independent components would improve clarity and estimability.
3. **Inclusion of Technical and UI/UX Details:** Many user stories lacked the necessary technical details or UI/UX specifications. For example, in the E-learning domain, the *User Story 4.1*, a story about grammar exercises did not specify whether they should be multiple-choice or fill-in-the-blank questions. Including such details in user stories would provide clearer guidance for developers and ensure alignment with user expectations.
4. **Alignment with INVEST Criteria:** While many user stories adhered to aspects of the INVEST criteria, this alone did not ensure they were fully prepared for development. Stories often met the structural requirements but still lacked critical elements such or contextual information needed for implementation. This highlights that while INVEST serves as a useful guideline, additional refinement is necessary to make user stories truly actionable for developers, reducing ambiguity and facilitating a smoother development process.

4.3 Recommendations for Improvement

Based on these findings, the following recommendations can help improve user story quality in agile development:

- **Develop Clear and Testable Acceptance Criteria:** Each user story should define explicit, testable conditions for completion. This would reduce ambiguity and ensure that developers know exactly what is expected.
- **Break Down Large Stories into Smaller, Independent Tasks:** By dividing complex stories into smaller, manageable units, teams can improve the clarity and estimability of stories, reducing cognitive load during development sprints.
- **Include Technical and UI/UX Specifications:** Where relevant, user stories should include or reference technical documentation, such as API specifications or database schemas, and provide UI/UX mockups to ensure consistency in implementation.

These recommendations align with previous research on the importance of clear requirements and practical user story frameworks in improving the efficiency and effectiveness of agile development processes.

4.4 Developer Feedback

The developer questionnaire provided key insights into the common challenges faced when working with user stories. The results indicate that 96.4% of developers have encountered issues with clarity or specificity in user stories, often leading to misunderstandings and rework. Additionally, 67.8% of respondents reported that user stories frequently lack essential technical details, such as data formats, API references, or tool-specific guidelines.

A critical issue identified was the absence of well-defined acceptance criteria, with 85.7% of developers strongly agreeing on their importance for determining when a story is complete. Developers also highlighted vague acceptance criteria (71.4%) and lack of clarity (69.6%) as the most significant challenges, followed by dependencies between stories (58.9%) and insufficient technical details (51.8%).

Nearly 95% of respondents agreed that including practical examples would improve user story comprehension and implementation, helping to clarify expected behaviors and avoid misinterpretation.These insights, along with a detailed breakdown of developer responses, are summarized in Table 4.

Table 4: Summary of Developer Feedback on User Stories.

Question	Responses
Have you ever faced difficulties with the clarity or specificity of user stories?	Yes: 96.4% No: 3.6%
Do user stories generally include sufficient technical details?	Strongly agree: 1.8% Agree: 30.4% Disagree: 60.7% Strongly disagree: 7.1%
Are clear acceptance criteria essential for effective user stories?	Strongly agree: 85.7% Agree: 14.3%
What are the biggest challenges when working with user stories?	Vague acceptance criteria: 71.4% Lack of clarity: 69.6% Lack of technical details: 51.8% Dependencies between stories: 58.9% No user value: 1.8% Breaking down large stories: 1.8%
Would practical examples improve the understanding of user stories?	Strongly agree: 50% Agree: 44.6% Disagree: 5.4%

These findings reinforce the need for clearer acceptance criteria, improved technical details, and the inclusion of practical examples to enhance the quality of user stories. The complete questionnaire results, along with detailed developer responses, are available in the technical document: Supporting Documentation for User Story Analysis.

5 CRITICAL ANALYSIS

The findings from the user story analysis and developer feedback revealed recurring issues in agile development, including vague descriptions, insufficient technical details, and unclear acceptance criteria. These challenges often cause delays, miscommunication, and rework, limiting the effectiveness of user stories despite frameworks like INVEST.

To address these issues, a Practical Guide for Writing Effective User Stories was developed. The guide highlights three essential practices:

Defining Clear Acceptance Criteria: Ensuring user stories include specific, measurable, and testable criteria for completion. Providing Technical Details: Including relevant specifications such as API endpoints, data formats, and expected system behavior. Breaking Down Complex Stories: Dividing large stories into smaller, independent tasks for easier implementation and estimation. These practices bridge theoretical principles and real-world development needs, enabling teams to improve user story quality and promote collaboration.

Figure 2 summarizes these recommendations, serving as a reference for teams aiming to enhance their requirements management process.

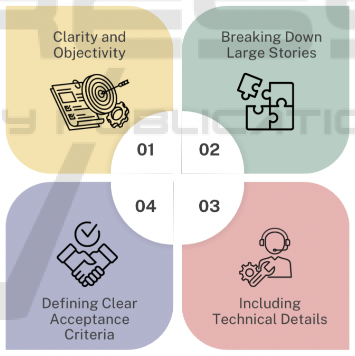
Clarity and Objectivity
Breaking Down Large Stories
01
02
04
03
Defining Clear Acceptance Criteria
Including Technical Details

Figure 2: Key areas of the Practical Guide for Writing Effective User Stories.

5.1 Clarity and Objectivity

One of the most prominent issues identified is the lack of clarity in user stories, as reported by over 90% of developers surveyed. Vague or ambiguous user stories lead to misinterpretations, resulting in rework and delays. For user stories to be effective, they must be written in a clear and straightforward manner, accessible to all stakeholders, including developers, testers, and product owners. This is particularly crucial in complex domains, such as e-commerce and security, where a lack of clarity can cause significant disruption.

To ensure clarity, user stories should be written with simple language and free of unnecessary technical jargon. Every term used should be well understood by all team members involved, and the goal of the story should be sharply focused on addressing a specific user need.

Example of a Clear User Story:
"As a registered user, I want to receive an email notification after placing an order, so that I can track my purchase status."

This example avoids ambiguity, making the required functionality easy to understand and implement, ensuring smooth communication between all involved parties.

5.2 Breaking Down Large Stories

Overly complex user stories that attempt to cover multiple actions—such as creation, update, and deletion in one—were reported by developers as difficult to estimate and test. These complex stories can overwhelm developers and result in missed deadlines or incomplete implementations. A recommended approach is to break down these large stories into smaller, independent tasks that deliver incremental value to the user, in line with the INVEST principles.

By ensuring that each smaller story remains estimable and testable within a sprint, the workflow improves, leading to more predictable progress and fewer bottlenecks in development.

Example of a Simplified Breakdown:
Instead of:
"As an admin, I want to manage users, including creating, updating, and deleting accounts."
Break it down into:
- *"As an admin, I want to create new user accounts to manage system access."* - *"As an admin, I want to update user accounts to reflect new information."*
- *"As an admin, I want to delete user accounts when necessary."*

This division allows each functionality to be developed and tested separately, making the process smoother and more efficient.

5.3 Defining Clear Acceptance Criteria

Another major issue highlighted by developers was the lack of well-defined acceptance criteria, with over 90% agreeing that this absence causes confusion during development. Without clear criteria, developers and testers struggle to determine when a user story is complete, leading to inconsistent results and potential delays.

Acceptance criteria must be objective, specific, and testable. They should define clear conditions that indicate when the functionality has been successfully implemented.

Example of Acceptance Criteria:

- The system must send a confirmation email upon successful order completion.
- The email must include the order number and purchase details.
- If the payment fails, the user must receive a notification with instructions to retry.

By including precise acceptance criteria like these, all stakeholders gain a shared understanding of what constitutes a successful implementation, reducing the risk of ambiguity or misinterpretation.

5.4 Including Technical Details

One of the most consistent findings from the developer feedback was the frequent lack of technical details in user stories. More than 75% of developers cited that user stories often did not include enough technical context—such as data formats, API details, or security requirements—resulting in delays and rework.

User stories should strike a balance between focusing on the user's needs and providing developers with the technical details they require for implementation. Where necessary, stories should reference supplementary technical documents, such as API documentation or database schemas, to provide clarity on implementation requirements.

Example:
If a user story involves integrating with an external API, it should include details such as expected data formats, HTTP methods, and response codes, or reference a technical document outlining these aspects.

5.5 Using Practical Examples

Nearly all developers (98%) agreed that practical examples significantly enhance their understanding of user stories. By providing clear examples that illustrate expected behaviors in different scenarios, developers can better grasp the requirements and avoid ambiguity during implementation.

Example:
"If the user selects express shipping, they should see an estimated delivery time within 24 hours."

Including practical examples like this provides concrete reference points for developers and helps align their work with stakeholder expectations.

5.6 Proposing Practical Solutions

The analysis highlights several key areas where improvements in the writing and structuring of user stories can directly address the challenges identified. Focusing on the following strategies can help agile teams develop clearer, more actionable stories:

- **Prioritize Clarity:** Write user stories using straightforward language, avoiding technical jargon where possible, and clearly defining the story's objective.
- **Break Down Complex Stories:** Divide large, complex stories into smaller, independent units that can be completed and tested within a sprint.
- **Define Specific Acceptance Criteria:** Ensure every user story includes clear, testable criteria that stakeholders can use to confirm completion.
- **Include Relevant Technical Details:** Provide or reference all necessary technical information, such as API documentation, database schemas, or flowcharts, to ensure developers have the information they need.
- **Use Practical Examples:** Incorporate examples into user stories to clarify how the system should behave in different scenarios.

By focusing on these improvements, agile teams can significantly reduce misunderstandings, improve efficiency, and deliver higher-quality software that better aligns with stakeholder expectations. These practical solutions reflect the combined insights of both the user story analysis and developer feedback, offering a path toward more effective user stories and smoother development processes.

6 CONCLUSION AND FUTURE WORK

This study has underscored the critical importance of improving user story writing practices in agile software development. The findings reveal that, while frameworks such as INVEST provide solid guidelines, their practical implementation often falls short in areas like clarity, well-defined acceptance criteria, and sufficient technical details. These shortcomings lead to confusion and inefficiencies that directly impact the development process.

The research conducted through both the user story analysis and developer feedback has highlighted several key areas of improvement:

- **Clarity and Simplicity:** User stories must be written in clear, accessible language that avoids ambiguity, ensuring all stakeholders understand the requirements.
- **Breaking Down Large Stories:** Large, complex stories should be divided into smaller, manageable units, each with independent value, to improve estimability and facilitate smooth sprint planning.
- **Clear Acceptance Criteria:** Explicit and testable acceptance criteria should be defined for each story, providing clear guidelines for developers and testers to know when the work is considered complete.
- **Inclusion of Technical Details:** User stories should include or reference the necessary technical specifications—such as data formats, API documentation, and security requirements—to reduce ambiguity and streamline the implementation process.
- **Use of Practical Examples:** Incorporating examples of expected outcomes, edge cases, and common scenarios helps clarify functionality and reduces miscommunication during implementation.

The suggestions put forth aim to create more actionable and reliable user stories, ultimately leading to better alignment with stakeholder expectations and improved development efficiency. The inclusion of additional supporting documentation, such as flowcharts, UI/UX prototypes, and technical diagrams, was also recommended by developers to further enhance story clarity.

For future work, we highlight two key areas for further exploration. First, validating these recommendations through real-world case studies would provide empirical evidence of their effectiveness in improving user story quality. By applying the proposed improvements in live software projects and analyzing their impact on development efficiency and stakeholder satisfaction, teams can better understand the practical benefits of enhanced user story practices.

Second, an interesting avenue for future research is leveraging AI tools to assist in refining and ensuring the quality of user stories. AI-driven approaches could help identify ambiguities, suggest refinements, and generate acceptance criteria automatically, reducing cognitive load for development teams and promoting more consistent story writing practices.

The quality of user stories has a direct impact on the success of agile software development. This study has demonstrated that clear language, well-defined acceptance criteria, sufficient technical details, and active collaboration between teams are essential elements for creating effective user stories. By implementing the practices recommended in this research, development teams might expect improved efficiency, fewer misunderstandings, and higher-quality software deliveries that meet stakeholder expectations.

As teams evolve their practices, user stories will become an even more powerful tool for managing requirements and ensuring smooth communication throughout the development process. Continued studies and experimentation will further refine the process of writing user stories, contributing to stronger alignment between technical teams and business objectives.

ACKNOWLEDGEMENTS

Our gratitude goes to the participants of this study, whose engagement and insights were fundamental to this research. We would like to thank the financial support granted by CNPq (314797/2023-8; 443934/2023-1; 445029/2024-2).

REFERENCES

Cohn, M. (2004). *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional.

Heck, P. and Zaidman, A. (2018). A systematic literature review on quality criteria for agile requirements specifications. *Software Quality Journal*, 26(1):127–160.

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.

Leffingwell, D. (2010). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Addison-Wesley Professional.

Leffingwell, D. (2018). *SAFe 4.5 Reference Guide: Scaled Agile Framework for Lean Enterprises*. Addison-Wesley Professional.

Lucassen, G., Dalpiaz, F., van der Werf, J. M. E., and Brinkkemper, S. (2015). Forging high-quality user stories: Towards a discipline for agile requirements. In *2015 IEEE 23rd International Requirements Engineering Conference (RE)*, pages 126–135. IEEE.

Lucassen, G., Dalpiaz, F., van der Werf, J. M. E., and Brinkkemper, S. (2016). Improving agile requirements: the quality user story framework and tool. *Requirements Engineering*, 21(3):383–403.

Wake, B. (2003). Invest in good stories, and smart tasks.

Williams, L. and Cockburn, A. (2003). Agile software development: it's about feedback and change. *Computer*, 36(6):39–43.