JOINETECH

The Implementation of Human-Centered Methodologies in Test-Driven Development: A Practical Example through Value Sensitive Design and Design Thinking

Luca Famà.
Corresponding author: Luca Famà, Department of Engineering, University of Messina, Italy. Email: luca.fama@studenti.unime.it

Antonio Cimino.
Department of Engineering, University of Messina, Italy. Email: antonio.cimino@unime.it

Vincenzo Corvello.
Department of Engineering, University of Messina, Italy. Email: vincenzo.corvello@unime.it

Citation:
Famà, L., Cimino, A., & Corvello, V. (2026). The Implementation of Human-Centered Methodologies in Test-Driven Development: A Practical Example through Value Sensitive Design and Design Thinking. ***JOINETECH,*** 2(1), 57–68. https://doi.org/10.65479/joinetech.38

ARTICLE INFO

Keywords: Test-Driven Development; Value Sensitive Design; Design Thinking; Software Development; Community Testing.

JEL codes: M15; O32; O33; O36.

ABSTRACT

Test-driven development (TDD) is a software development approach that has become established within agile techniques. It enables the improvement of technical project performance by ensuring rapid feedback between testing and development. The literature on TDD has so far focused predominantly on the technical aspects of the process, overlooking the human dimension; this highlights a clear research gap concerning the potential integration of complementary methodologies into TDD that could enhance its overall effectiveness. The aim of this action research study is to analyze whether and how value sensitive design (VSD) and design thinking (DT) can be used to improve TDD, allowing the identification of users' values and enhancing their creativity from the earliest stages of software testing. To this end, the findings of the research directly guided the design and development of Open Test 2.0, a digital platform created to support TDD while integrating the proposed methodologies. The platform includes several modules that allow testers to explore aspects related to values and creativity through tools such as communities and dedicated questionnaires. Although the platform is not yet accessible to external users and external feedback is therefore not available, the results from questionnaires distributed among the involved stakeholders still suggest that TDD can be supported by complementary methodologies, specifically VSD and DT, which help address some of its existing gaps.

Submission: March 15, 2026, Acceptance: April 20, 2026. Published: May 2026.

1. Introduction

In recent decades, the software development process has undergone a profound evolution aimed at ensuring greater flexibility, adaptability, and speed in the creation of digital products. This transformation has led to the progressive abandonment of rigid and sequential models such as the so-called waterfall model, characterized by a clear division into successive and minimally interactive phases, in favor of agile approaches that are more iterative and focused on delivering value to the end user (Dima and Maassen, 2018).

In this cultural and operational context lies test-driven development (TDD), a programming practice that overturns the traditional development approach by inverting its phases, almost counterintuitively: Instead of writing the code first and then verifying its behavior, development begins by writing tests that define the expected behavior of the system, and only afterward is the minimal code necessary to satisfy them implemented (Munir et al., 2014).

It clearly emerges that, despite all the technical advantages it provides, such as improved internal and external software quality (Beck, 2003), greater ease of code maintenance (Parsa et al., 2025), and a smoother evolution of the software over time (Siniaalto and Abrahamsson, 2007), TDD focuses primarily on code production, without systematically addressing the role of the end user in the development process. This highlights a significant research gap concerning the integration of user-centered perspectives within TDD practices. In a contemporary context in which software is increasingly expected to meet complex, shifting, and highly context-dependent needs, it becomes essential to shift attention toward the overall design of the experience and toward values. It is therefore fundamental to question how the user is involved in crucial development moments not only as a validator of functionalities but as a co-designer of meanings, needs, and priorities, evaluating, for example, the psychological impact of the software and its inclusiveness or proposing improvements in its structure and organization.

In this perspective, the object of this research is the creation of a software architecture that enables the systematic implementation of human-centered methodologies within the TDD process.

Accordingly, the following research question guides this work: How can human-centered methodologies be integrated into the process of a TDD-based software platform?

To address this research question coherently and thoroughly, it is necessary to examine approaches that enable me-

aningful and active user participation. A first central aspect concerns values: Users' sensitivities and principles cannot be overlooked, as they represent a fundamental reference point for designing truly effective and shared solutions. In this sense, value sensitive design (VSD) offers useful tools for identifying and understanding user values, integrating them into the design process (Friedman et al., 2012).

A second element often neglected by TDD is creativity, understood as the user's ability to contribute actively to software development by proposing new functionalities or modifying existing ones. This aspect is addressed through design thinking (DT), a methodology that fosters collaborative design and the generation of innovative solutions starting from users' real needs (Serrat, 2017). More specifically, this study focuses on the integration of VSD and DT within the TDD process.

The two dimensions considered, values and creativity, significantly expand the possibilities for intervention beyond TDD, offering concrete ways to involve the user in the definition and evolution of software. However, the application of such methodologies to TDD is, so far, relatively rare. To bridge this gap, it is not enough to reflect on a theoretical level; it is also necessary to test these ideas in practice through real, concrete implementations. To verify the possibility of bringing together the technical quality objectives of the code with user satisfaction and involvement, these concepts were applied to a real project: Open Test 2.0, which served as the operational context for experimenting with the integration of the two identified methodologies.

The project aims to develop an innovative framework for community testing of software solutions, placing at its center the collaboration among local institutions, associations, and communities of super users. The development of this platform is part of a research project whose team, following a precise division of tasks and responsibilities, contributed its specific expertise. In particular, the work of the scientific group focused on a preliminary analysis of the state of the art of the methodologies under examination and on the development of the platform's architecture, defining its objects, tools, roles, and processes. This approach allows the platform to concretely embody the proposed integration between TDD and human-centered approaches. The framework goes beyond traditional testing, aiming to create a truly interactive and collaborative environment.

The main contribution of this study therefore lies in the development of this platform, which serves as a concrete example and good practice for the implementation of TDD enriched with human-centered methodologies.

The final objective is to generate a dynamic and participatory environment in which cooperation between users and developers becomes a key factor for innovation and continuous improvement of the tested solutions.

To present this study comprehensively, the remainder of this article is organized as follows: Section 2 provides a review of the literature, focusing on TDD and the limitations related to user involvement, as well as introducing VSD and DT as complementary approaches. Section 3 describes the methodological approach adopted, outlining the action research process and the development of the Open Test 2.0 platform as an experimental context. Section 4 presents the results, detailing the architecture of the framework and the implementation of VSD and DT within the platform through its tools and collaborative environments. Section 5 discusses the main findings, analyzing how the integration of human-centered methodologies within TDD contributes to a more participatory, value-oriented and creative software development process. Finally, Section 6 concludes the paper by summarizing the main contributions, highlighting the limitations of the study and outlining possible directions for future research.

2. Literature Review

2.1 The Importance of the Testing Phase and Test-Driven Development

The increasing complexity of software systems entails greater responsibility in ensuring their reliability and security (Wang et al., 2024). Software malfunctions can in fact lead to serious consequences, ranging from economic loss to risks for human safety and even human life (Ahsan and Anwer, 2024). In this scenario, the discipline of software engineering plays a crucial role, covering the entire software life cycle from design to maintenance with the objective of optimizing each phase and minimizing risks (Wohlin et al., 2012).

One of the most relevant phases of the software life cycle is software testing (ST), which is concerned with verifying and validating the quality, reliability, and safety of the product before its release. Through testing, it becomes possible to identify defects, errors, and vulnerabilities that could compromise the functioning of the software in real environments (Young, 2008).

The strategic use of testing during software design and development offers multiple advantages: On the one hand, it enables the development team to work more efficiently (Augusto, 2020), reducing time and costs; on the other, it improves the end-user experience by delivering a higher-quality product thereby increasing user satisfaction and loyalty (Büyükyumukoğlu et al., 2017).

However, there is no single ideal way to approach testing: every project has its own specific characteristics and therefore requires the adoption of the methodology best suited to the objectives to be achieved. Among these methodologies is Extreme Programming (XP) (Beck and Fowler, 2000), characterized by an agile approach that integrates testing into the development process through short iterations and continuous feedback. Every section of code is tested and validated before being considered complete, ensuring responsiveness to change and high software quality. XP is particularly suitable for contexts in which user requirements are subject to frequent updates and modifications (Drobka et al., 2004).

Among the 12 practices of extreme programming is TDD, which is based on two principles:

- Test-first programming
- Continuous refactoring

The principle of "test-first" is based on the idea that tests should be written before production code, while refactoring refers to the continuous improvement of the code. This approach is considered one of the core practices of agile development and guides the process through a cyclical sequence known as red–green–refactor:

- Red phase: Development of a new feature begins by writing an automated test that defines its expected behavior, requiring creative effort from the programmer since the functionality does not yet exist. The test must be executable and must fail, demonstrating that the logic has not been implemented. The red phase ends when the new test compiles, executes correctly, and fails as expected, marking the starting point for the subsequent phases (Beck, 2022).
- Green phase: In this phase, the developer writes only the minimal amount of code needed to make the failing test pass, without worrying about final code quality. Once the solution is implemented, the developer executes the entire test suite to ensure that no regressions have been introduced. The green phase concludes when all tests pass (Beck, 2022).
- Refactoring: After all tests have passed, the developer begins refactoring to improve the internal structure of the code without modifying its behavior. Through small interventions, such as removing redundancies, simplifying complex parts, and reorganizing classes, the quality of the software increases. After each modification, the full test suite is run again to ensure that no regressions have been introduced (Beck, 2022).

A further set of essential characteristics of the TDD process concerns granularity, uniformity, and refactoring effort (Fucci et al., 2017):

- Granularity refers to the duration of individual development cycles, emphasizing the importance of short and frequent steps.
- Uniformity describes the consistency in the duration of these cycles over time, reflecting a systematic and regular approach.
- Refactoring effort represents the amount of restructuring performed by the developer, highlighting the crucial role of this practice in maintaining internal software quality.

From this initial description of TDD, it is clear that none of its phases focuses on aspects beyond code quality/functionality, thus highlighting both strengths and weaknesses of the method. Many authors consider TDD highly effective for obtaining clean and functioning code, emphasizing how it makes expected behaviors explicit, guides interface design, and supports iterative and incremental development (Beck and Fowler, 2000).

The literature frequently associates TDD with various significant benefits, such as:

- Improved internal and external software quality (Beck, 2003).
- Increased test coverage.
- Greater ease of code maintenance.
- Effective support for continuous refactoring (Parsa et al., 2025).

In industrial contexts, several developers report fewer production bugs, faster development cycles, and a noticeable increase in productivity especially when dealing with consolidated codebases or well-defined functionalities. TDD can also reduce the fear of regressions when adding new features, thereby improving perceived development safety and facilitating a smoother evolution of the software over time (Siniaalto and Abrahamsson, 2007).

Despite the numerous theoretical and practical advantages, the effectiveness of TDD remains a subject of debate within the scientific community (Aniche and Gerosa, 2015); multiple empirical studies have analyzed its impact on quality and productivity (Tosun et al., 2017), but the results are often conflicting and strongly dependent on the context, project complexity, and developers' experience.

For example, a longitudinal study conducted on 30 junior developers showed that, although TDD did not significantly improve software quality or productivity in the short term, it led to the creation of a greater number of tests with high defect-detection capability. Furthermore, the practice was retained over time, suggesting that TDD may exert a positive influence on development culture and quality-oriented attitudes (Baldassarre et al., 2021). Another experiment, conducted with 24 developers in an industrial setting, found that TDD improves productivity for simple tasks but tends to slow down teams when working with complex tasks (Tosun et al., 2017). Again, no significant differences emerged in software quality compared with the test-last approach, highlighting how task complexity strongly affects the effectiveness of TDD.

One of the most frequent criticisms concerns the cost and difficulty of writing valid and meaningful tests before the code, especially in scientific or highly complex projects. In such contexts, it is often difficult to anticipate all test scenarios in advance, and the lack of adequate tools or specific expertise can become an obstacle to the effective adoption of TDD (Fucci et al., 2015). Furthermore, TDD shows limitations in terms of user involvement, neglecting aspects related to values (Sharma et al., 2023) and creativity as a co-design tool (Díaz and Aedo, 2020). The focus is strongly placed on tech-

nology as the starting point of the design process: Development is guided by technical possibilities rather than by the needs, experiences, or values of the people involved. This orientation can lead to technically high-performing solutions that, however, are poorly aligned with users' values or are insufficiently meaningful to them. This is particularly relevant given that the creation of user-centered approaches in development processes is one of today's major challenges (Szabó and Hercegfi, 2022). Participatory creativity, meaning co-design moments and collective exploration of ideas, also tends to be marginalized, reducing the potential to generate more inclusive, sustainable, and socially grounded solutions. This highlights a research gap concerning the systematic integration of participatory and value-oriented approaches within technically driven development practices.

In light of these considerations, two methodologies have been identified that place specific emphasis on ethical-value aspects and the creative dimension of the design process, VSD and DT, with the aim of assessing their compatibility with TDD.

2.2 Value Sensitive Design

VSD is an approach aimed at placing human values and ethics at the center of the design process of a new technology (Gerdes and Frandsen, 2023).

Initially conceived for applications in the field of information systems, VSD is now adopted across a wide range of sectors. In the specific case relevant to this study, we analyze how VSD can be applied to software development, with particular attention to the software testing phase. This method, which aims to support designers in considering ethical values, does not present itself as a rigid system prescribing what is right or wrong; rather, it proposes an approach that encourages reflection on sensitive issues, generating outcomes that may vary depending on the situation and specific context.

Table 1 delineates a set of fundamental values that have been identified as critical considerations in the development of emerging technologies (Friedman and Hendry, 2019).

Table 1. Set of values.

Value	Definition
Human rights	Inalienable and fundamental rights to which all people are entitled
Social justice	Includes both procedural justice (the process is fair) and distributive justice (the outcomes are fair)
Human well-being	The physical, material, and psychological well-being of individuals
Accessibility	Enabling all users to participate in the success of the technology
Respect	Treating people with consideration and valuing their perspective
Calmness	Fostering a peaceful and composed psychological state
Freedom from bias	Avoiding systematic differential treatment among individuals or groups, including pre-existing social biases, technical biases, and emerging social biases
Ownership	The right to own an object (or information), use it, manage it, derive income from it, and transfer it as inheritance
Privacy	An individual's claim or right to determine which personal information about themselves can be shared and used by others
Trust	Expectations that interactions between people occur in good faith or imply goodwill and do not aim to exploit vulnerabilities or dependencies
Accountability	Ensuring that the actions of a person, a group, or an institution can be traced back to them, both causally and in terms of responsibility attribution
Autonomy	The ability of individuals to decide, plan, and act in ways they believe will help them achieve their goals, as well as the ability to choose their own goals
Informed consent	Gathering people's consent, including criteria of disclosure, understanding, voluntariness, and competence
Identity	The understanding individuals have of themselves over time and their capacity to represent themselves that way to others
Environmental sustainability	Maintaining the integrity and stability of ecological systems, processes, and components to meet present needs without compromising the ability to meet future ones

Source: Friedman and Hendry, 2019.

These values are not proposed as absolute, since depending on the reference context, new ones may be added, or some of those listed above may become less central. The intention is to provide guidelines for the application of the VSD method. To incorporate human values into the design and development of a new technology, VSD suggests a tripartite, integrative, and iterative methodology (Friedman et al., 2006); this method is based on continuous questioning of the ethical issues related to a given technology (Manders-Huits, 2011), generating partial solutions that can later be revisited and modified as the results of the process provide further insights and information. The three investigations of the methodology are conceptual (value-focused), empirical, and technical (Manders-Huits, 2011).

A conceptual, or value-focused, investigation aims to analyze the information that emerges from the empirical investigation and identify all the ethical and value-related issues that may be relevant to the development of the new technology (Davis and Nathan, 2015). These issues are then summarized and translated into practice during the technical investigation.

VSD is characterized by its flexibility and iterative nature within the design process. Although each of the three investigations focuses on a specific aspect, shared features emerge that connect them. Among these, the constant analysis of ethical issues and the attention to the objectives defined in

the initial phase are particularly significant. The three investigations should be carried out the following theoretical order: empirical, conceptual, and technical. However, this flow is not always followed, as shown by an analysis of 17 projects: 16 begin with a conceptual investigation, followed by 9 beginning with an empirical analysis, 7 proceeding to technical analysis after the conceptual one and concluding by empirically validating the values, and only 4 projects performing an iterative investigation across all three methodologies (Winkler and Spiekermann, 2021).

There are practices which, depending on the reference context, can help designers identify crucial aspects within ethical issues and analyze the perspectives of all stakeholders. Among these are value scenarios and value cards (Friedman et al., 2012). Other analogue tools also exist (Peters et al., 2020) and are designed to support collaborative and value sensitive design. They promote cooperation and, above all, allow complex problems to be visualized in a tangible way, bypassing the limitations of certain digital tools. These include board games, posters, wearable simulation devices, construction materials, and others.

2.2. Design Thinking

The current market forces anyone involved in the development of new technologies to focus on innovation (Dabić et al., 2023), understood as the ability to generate something new for which someone is willing to pay, something capable of creating interest, or the reorganization of an existing product or service by presenting it in a different form. Today, the ability to innovate represents perhaps the greatest competitive advantage for those who are able to cultivate it: the ability to keep pace with change and identify users' needs (Arsawan et al., 2022). Design is typically viewed as a downstream phase of a process, but nowadays, already during the design phase, it is crucial to begin considering design aspects to improve the user experience.

DT therefore places people at the center, and to do so, it is necessary to be highly empathetic and capable of interpreting consumers' desires (Serrat, 2017).

It develops through three main phases: inspiration, ideation, and implementation (Fraser, 2009). These phases are characterized by an iterative cycle that allows the design team to continuously review proposed solutions. Throughout the process, questions are asked, and new ideas are generated, fueling creativity and the search for increasingly accurate solutions. This approach enables the organization of the flow of ideas in a more structured way, facilitating the understanding of problems and opportunities. DT is not merely a problem-solving process but a continuous exchange between theory and practice that stimulates innovation.

More specifically, it is a prototyping process during which all necessary elements are added step by step to arrive at a final product that meets the initial requirements as closely as possible. In the specific context of software development, design thinking guides the team through phases of design, testing, and successive improvements, ensuring that the final product not only meets user expectations but also addresses the challenges that emerged during the process (Serrat, 2017). In summary, it helps build practical and innovative solutions, always centered on the user's experience and needs, through an iterative and reflective approach.

Three mechanisms interact with each other in the application of design thinking (Serrat, 2017):

- Deep understanding of the user: The first step is the ability to shift one's perspective by putting oneself in the shoes of the user and other stakeholders. This is necessary to achieve a 360° view of the issue under discussion and to identify the needs, expectations, and motivations of those involved in the process.

- Concept visualization: Creativity plays a central role in this phase, becoming essential for the development of new solutions. There are no constraints, only opportunities.

- Strategic business design: The final mechanism is responsible for making the previously collected information concrete and transforming all the ideas that emerged during brainstorming or process investigation into feasible and optimal solutions.

3. Methodology

This article aims to investigate the design and development of a software architecture that enables the systematic integration of human-centered methodologies within the TDD process. Particular attention is given to the interaction between VSD and DT, analyzing how these approaches can be incorporated into TDD practices. The main objectives are to assess the compatibility of these methodologies with testing activities, to explore how they can be effectively embedded within a structured software development process, and to evaluate whether their combined application can produce tangible benefits in terms of efficiency, effectiveness, and overall software quality.

To analyze these aspects empirically rather than only theoretically, it was decided to actively participate in the real project Open Test 2.0, contributing directly to its development. The study adopts an action research approach, in which researchers are directly involved in the design and implementation process, enabling a continuous interaction between theory and practice. This made it possible to closely observe design dynamics, concretely apply the methodologies under study, and collect data useful for critically assessing their effects on the entire development cycle. Through this direct experience, it was possible to evaluate both the strengths and potential weaknesses of integrating traditional and innovative approaches, thereby offering practical insights for their adoption in future design contexts.

Open Test 2.0 is an innovative platform designed to improve the way software testing activities are conducted, through an approach based on the active participation of multiple

stakeholders. Open Test 2.0 proposes a community testing model, meaning a system in which the testing of digital solutions is not limited to developers or internal teams, but extends to a network of users made up of local institutions, associations, companies, and other organizations. These actors are involved not only as simple executors of tests, but as true co-protagonists in the evaluation process through defining requirements, identifying critical issues and participating in the co-design of solutions. The framework is based on a set of methodological principles that include VSD and DT, with the aim of promoting shared, value-oriented, and strongly collaborative innovation. The adoption of VSD makes it possible to integrate not only technical functionalities but also the values, needs, and expectations of the users involved into the development process, while DT encourages collaborative design among all platform users. This hybrid approach represents a significant evolution compared with traditional testing models, as it increases stakeholder involvement across all phases of the software life cycle. Another distinctive element is its flexible and collaborative organizational structure: within the framework, each participant can operate autonomously (within their own organization) or join collaborative communities, open groups formed around specific areas of expertise or testing projects. These communities create operational synergies, facilitating the sharing of resources, techniques, and existing knowledge among members.

Participation in Open Test 2.0 provided a concrete opportunity to test the methodologies examined in this study and made it possible to observe the interaction between theoretical approaches and design practice, offering an empirical basis for evaluating the compatibility and effectiveness of integrating TDD, VSD, and DT in a real context. The experience offered valuable data and insights for understanding how the combination of these tools can positively influence software project quality and support the adoption of participatory models in technological innovation.

To actively contribute to the development of the research project, extensive preliminary literature analysis was conducted, focusing on the main topics under investigation: VSD, DT, and TDD. The review was carried out using the Google Scholar search engine, selecting the most relevant scientific articles on the basis of content and thematic alignment. Particular attention was given to publications in high-impact scientific journals, including those of Elsevier, Taylor & Francis, Wiley, Emerald, Springer, and other internationally recognized sources known for their reliability and rigor.

During the active working period of the project, periodic meetings were organized on a monthly basis and, during more intensive phases, even biweekly. All actors involved in the development of the platform participated in these meetings. These moments of discussion were fundamental for ensuring continuous alignment of the team regarding the state of progress of the project, allowing integrated and cross-functional discussion of the different areas involved—including scientific research, software development, and organizational and administrative aspects. The design of Open Test 2.0 did not follow a siloed approach, but rather a collaborative and cross-functional one, grounded in continuous dialogue among stakeholders with different professional backgrounds. This process facilitated the emergence of shared needs and the collective construction of a unified project vision, contributing to the development of a platform that is more coherent with collective expectations and better aligned with the real needs of end users.

Active contribution to the Open Test 2.0 project was not limited only to the literature review phase but also extended to the definition of the framework architecture. Once the project objectives had been clarified and the corresponding technical and functional constraints identified, work proceeded with the complete mapping of the platform's components, analyzing in detail their characteristics, functionalities, and interrelationships. This work made it possible to build a structured and systemic vision of the framework, which in turn guided the subsequent stages of development.

The outcome of this phase was formalized in a deliverable, created with the aim of providing the development team with a clear and shared design reference. This document served as a fundamental operational support for the technical implementation of the platform, ensuring coherence between the conceptual vision and the engineering decisions adopted during the development phases.

To evaluate the results of the study, data were collected through a structured questionnaire distributed among the participants involved in the Open Test 2.0 project. The questionnaire was designed to capture perceptions regarding the effectiveness of integrating TDD with VSD and DT, as well as to assess aspects related to usability, collaboration, and overall satisfaction with the development process. The collected data provided an initial empirical basis for evaluating the impact of the proposed approach and for identifying both its strengths and potential areas for improvement. However, as the platform is currently still in a prototyping phase, it has not yet been possible to gather feedback from external users, testers, and participating organizations. This represents a limitation of the study, as the evaluation is primarily based on the perspectives of actors directly involved in the project, rather than on evidence derived from real-world implementation and broader user engagement.

4. Results

The architecture of the Open Test 2.0 platform was developed by considering four fundamental elements: objects, tools, roles, and integrated processes. Their interaction made it possible to create a virtual ecosystem that meets the needs and objectives defined during the design phase.

Thanks to the schematic breakdown presented in Fig. 1, it is possible to understand the overall organization of the platform, thus providing an overview of the project under analysis.

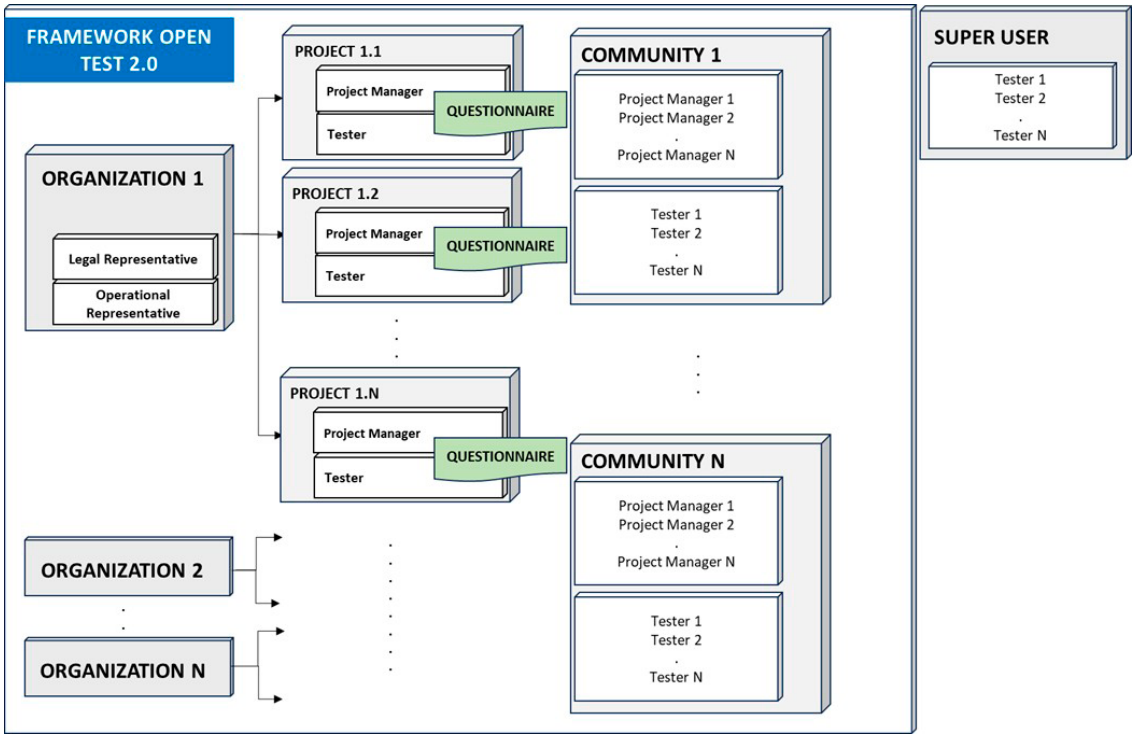


Fig. 1. Platform framework of Open Test 2.0. Source: Authors' own creation.

The structure is designed to intuitively distinguish all elements of the platform through the use of colors, enabling easy identification. Furthermore, to support clear recognition and contextualization, the elements are represented three-dimensionally.

On the lowest level, all the objects present in Open Test 2.0 are depicted, namely the virtual spaces where all activities take place. On each of these objects, thin parallelepipeds have been placed to highlight the actors operating within that specific object. The "questionnaire" block is represented in a two-dimensional format and colored green to emphasize that it is the tool used within the platform for testing operations.

For greater descriptive clarity, the details related to a single organization (object) and the *n* projects (objects) associated with it have been schematized. Already in this first part of the architecture, it is possible to identify all four roles present on the platform: legal representative, operational contact person, project manager, and tester. The questionnaire, as previously noted, is the tool used on the platform to carry out testing activities; for this reason, it has been represented on the "project" object and in direct contact with the actors involved.

On the right side of the framework are the communities (objects), which are the virtual spaces where all users on the platform can exchange information and opinions with the aim of creating new design proposals. This is where DT is implemented. Finally, external to the macrostructure described so far, there is the last object of Open Test 2.0, namely the super user: a legal entity to which a number *n* of testers belong.

Table 2 provides a detailed presentation of all the elements that interact within the Open Test 2.0 platform.

Table 2. Overview of the Open Test 2.0 elements

Elements	Type	Definition
Objects	Organization Project Community Super user	The objects represent the basic elements of the platform. They are the virtual "places" within which users interact in their various forms.
Roles	Legal representative Operational representative Project manager Tester Platform manager	Roles define the tasks that each user can and must carry out within the platform.
Tools	Questionnaire Technical module Value module Evolutionary module	The tool used to carry out the tests is a questionnaire, which, when needed, is supplemented with one or more modules among those available.
Integrated process	Nonfunctional testing Value testing (VSD) Evolutionary testing (DT)	An integrated process is the set of activities carried out by the actors involved, starting from the creation of an organization and continuing through the implementation of the necessary changes identified during the testing phase.

Source: Authors' own creation.

4.1 Implementation of Value Sensitive Design and Design Thinking

The two methodologies, VSD and DT, were introduced within the platform in two elements: the tools (tests) and the "community" object. The analysis of the architecture will therefore focus on these aspects.

4.1.1 Tools

Within the architecture of the innovative framework developed for the Open Test 2.0 project, tools essential for enabling the actors involved to provide feedback on the projects under examination are included. There are several types of software testing, each with its own specific characteristics. In the Open Test 2.0 platform, an approach based on the tester's experience is adopted, requiring testers to identify any anomalies or misalignments between the desired result and the one actually achieved.

This type of test is particularly effective in contexts requiring subjective evaluations or in uncertain scenarios. Despite the flexibility of this approach, the use of these tools requires time and resources, with a greater likelihood of errors caused by distraction or superficiality during the analysis phase.

The tests proposed in the development of this framework fall within the following categories:

- Nonfunctional testing
- Value testing (VSD)
- Evolutionary testing (DT)

In the first case, testers are asked to verify the general characteristics of a software application. This type of testing focuses on *how* the software performs rather than *what* it does. The main categories of nonfunctional testing include security testing, performance testing, usability testing, and compatibility testing.

The VSD and DT tests aim to make the software under analysis more sensitive and aligned to users' values, and to involve testers in the design process, not only during the correction phase but also during creation.

All three types of tests are delivered to testers through a single tool: the questionnaire.

The questionnaires consist of a number (*n*) of questions to which testers respond on the basis of their experience accumulated during the inspection of the software under analysis. Within the Open Test 2.0 platform, there is a question database that can be used when needed. This allows those who create the questionnaire not only to propose questions in a standardized, high-quality form but also to save time during the design phase. Should it be necessary to insert questions that are not present in the database, it is always possible to create new ones from scratch.

The three types of tests identified in the previous paragraph are implemented through three modules, one for each type, which may exist individually or in combination within each questionnaire (a questionnaire may contain only one module, two of them, or all three).

The three modules are:

- Technical module
- Value module
- Evolutionary module

They are modular and can therefore be simplified or made more complex depending on the experience of the testers to whom they will be administered. For the testers' evaluations, a four-point Likert scale is used; the absence of a neutral midpoint encourages testers to take a position in one direction or the other. The available options are "not at all," "a little," "fairly," and "very."

The main characteristics of each testing module are presented in Table 3.

Table 3. Testing module characteristics

Module	Main features
Technical	Oriented toward the resolution of technical issues such as bugs or malfunctions in the software under analysis. Testers focus on functionality.
Value-oriented	Aimed at analyzing values and has the objective of identifying those not considered during the initial development phase. Testers focus on the ethical/emotional dimension. It implements VSD.
Evolutionary	Is the only module that includes open-ended questions, with the aim of stimulating new ideas and proposals about the software under analysis. It implements DT.

Source: Authors' own creation.

4.1.2 Objects

Objects may be concrete or conceptual elements that define the structure and functioning of the system, representing the pillars on which collaboration and testing processes are based. They constitute the fundamental building blocks through which the user experience is structured and shared value is created. Understanding how these objects work means having a clear view of the participatory logics and enabling mechanisms that characterize Open Test 2.0.

They shape the operational dynamics within the platform, supporting the sharing of resources, knowledge, and objectives among the various actors involved. Thanks to their interconnected nature, the objects make it possible to organize work in a structured manner, promoting an iterative and adaptive approach that facilitates continuous innovation. Moreover, they support the management of relationships be-

tween users, the definition of responsibilities, and the traceability of activities, contributing to making the system more efficient and responsive to the needs of the context in which it is applied.

Within the platform under analysis, four objects have been identified:

- Organizations
- Projects
- Communities
- The super user

In particular, communities represent collaborative spaces in which groups of users can interact, share experiences, and explore topics of common interest. They are less structured and thematic environments created to host discussions, technical questions, reports, or suggestions but above all to support the development of new design proposals. They are transparent digital spaces where every contribution is visible and accessible to all platform users. The strength of communities lies in their ability to concentrate collective intelligence around specific problems, promoting shared, traceable, and applicable solutions. Communities therefore become fundamental support tools that enable the implementation of DT within the Open Test 2.0 platform.

Specifically, it is the responsibility of the person who initiates a given discussion to gather all the information, suggestions, and design proposals received within the community and, with the support of the development team, complete the implementation phase on the software under analysis. Participation is always voluntary but encouraged by the opportunity to learn in a dynamic and open environment where users can collaborate freely, discuss working methodologies, and contribute to the collective growth of the platform ecosystem. Each user may actively participate in discussions, share documents, and initiate targeted conversations through integrated communication tools such as chat systems.

5. Discussion

At the core of the Open Test 2.0 project lies the awareness that software testing constitutes a crucial phase in the development of digital solutions. In a landscape increasingly oriented toward rapid and iterative software production, correctly testing what is being developed is essential to ensure the quality, security, and reliability of applications. Traditional testing models, which are technical and carried out by teams internal to the project itself, however, struggle to capture the complexity of real-world use and to include in the evaluation aspects related to the user experience, social values, or collaboration dynamics. It is in this context that the Open Test 2.0 proposal emerges: a participatory platform based on a community testing model, which aims to actively involve real users, super-users, and institutional actors in the software testing process. The objective is not only to identify bugs or functional anomalies but also to understand how the software is actually perceived, used, and valued in everyday usage contexts. For this to be possible, however, technical tools are not enough; a paradigm shift is needed, a design approach that places people, their needs, and their values at the center.

It is precisely from this need that the limits of TDD emerge, which, despite representing one of the fundamental practices of agile development and guaranteeing a high consistency between code and functional requirements (Beck and Fowler, 2000), focuses almost exclusively on the technical dimension of the software. Tests, in fact, are written before the actual code but remain confined to verifying the expected behavior in computational terms. The end user, in this scenario, has no active role either in defining the functionalities or in reflecting on what has value. TDD, therefore, is strongly dependent on the context and the experience of developers (Tosun et al. 2017) and on the degree of complexity of the task to be performed (Fucci et al., 2015); it does not allow for true user involvement in the design phase, does not contemplate the ethical dimension nor the reflection on values, and neglects the creative potential of those who will use or test the software. In other words, it is limited to answering the following question: “Does it work as expected?”

In the context of Open Test 2.0, it was therefore decided to experiment with the integration of two complementary approaches (VSD and DT), which could broaden the design horizon by including the human, creative, and motivational dimension in the software development process, responding to current challenges (Szabó and Hercegfi, 2022) and building on the positive influence of TDD toward a quality-oriented attitude (Baldassarre et al., 2021). From a practical perspective, this integration responds to the need for development teams to incorporate lightweight, user-centered tools within existing workflows, without disrupting established agile practices. The introduction of the two approaches into the project’s innovative framework proved to be concretely feasible and, in many cases, extremely useful. Although each of them responds to different objectives, their integration has proved effective in promoting a richer, more reflective, and more participatory design process. VSD was integrated from the earliest stages of the project, through stakeholder mapping and discussions on specific topics, such as the voluntary work carried out by testers within the platform. The most significant application of this approach materialized in the innovative framework. In particular, within the tools (questionnaires), a checklist of questions dedicated to values was included, allowing testers to express an evaluation of the ethical aspect of the different software analyzed; additionally, within the framework, a space (community) was created to encourage and facilitate ongoing discussion between testers and platform users on ethical and value-related topics. DT was implemented both within the questionnaires, with the inclusion of a set of questions that allow for open-ended answers, fostering collaborative design and the possibility of seeing testers’ proposals realized should the development team consider them interesting, and within the communities.

By doing so, the entire development process is no longer at the exclusive discretion of developers, but the role of the user becomes central in defining the characteristics and functionalities of the software. The path taken is perfectly consistent with the principles of agile development (Dybå and Dingsøyr, 2008) and moves in a direction aligned with current human-centered development trends, overcoming the problems related to TDD as studied in the scientific literature and applied in case studies. In this sense, the Open Test 2.0 model can be considered replicable in other development contexts, as it is based on modular elements such as participatory tools, value-oriented evaluation mechanisms, and collaborative environments that can be adapted to different organizational settings and levels of complexity.

From a theoretical perspective, the study contributes to extending the scope of TDD beyond its traditional technical boundaries, demonstrating how it can be enriched through the integration of human-centered methodologies and offering an initial framework for combining VSD and DT within software engineering practices.

At the same time, some limitations must be acknowledged: As the platform is still in a prototyping phase, it has not yet been tested by external users, testers, or organizations beyond the development team, limiting the possibility of validating the results in real-world contexts. Moreover, the action research approach, while enabling deep involvement in the project, may introduce potential biases, as researchers participated directly in both the development and evaluation phases; in particular, responses collected through the questionnaire may reflect subjective perspectives influenced by participants' proximity to the project.

6. Conclusions

This work examined the design and development of a software architecture aimed at enhancing TDD, known for its high efficiency in producing quality code, through the integration of design approaches more oriented toward creativity and ethics, namely VSD and DT. The objective was to explore how such methodologies can be systematically embedded within the TDD process to foster more active and meaningful user involvement, particularly during the testing phase.

The Open Test 2.0 project served as a testing ground for this conceptual and methodological integration, redefining the role of the tester, from a simple technical validator to a creative actor and ethical guide within the co-design process. Through the use of collaborative objects and tools (communities and questionnaires), Open Test 2.0 aims to promote participatory testing based on shared values. This approach enables not only the improvement of technical product quality but also the alignment of software with users' deeper needs, promoting forms of social as well as technological innovation. The study clearly shows that adopting methodologies such as DT and VSD can provide an effective framework for expanding the horizons of TDD, making the testing process more inclusive, empathetic, and motivating.

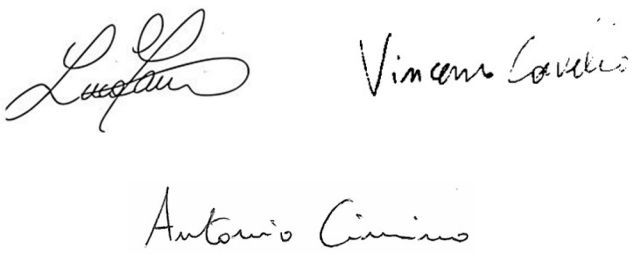
The work also highlighted several significant limitations: First, the Open Test 2.0 platform is still "open," and therefore, it was not possible to conduct an extensive performance evaluation; empirical data are currently lacking that would allow an assessment of the concrete effectiveness of the proposal in real and diverse contexts. The platform was also designed exclusively for web portals, a choice motivated by the need for focus, but which limits the scalability of the framework to other types of software. Looking ahead, it will be essential to conduct case studies, field experiments, and comparative analyses with other community testing tools to better understand the potential and areas for improvement of the proposed model.

In conclusion, the work carried out demonstrates that TDD, DT, and VSD are not alternative approaches but potentially synergistic ones and can be successfully integrated into a digital platform designed to enhance the active role of testers, including nonexperts, in the innovation process. Although some uncertainties remain regarding the practical applicability and scalability of the solution, this work represents a first step toward a broader and more participatory vision of software testing, one in which the value of code lies not only in its functional correctness but also in its ability to reflect the values, expectations, and desires of the people who will use it.

Acknowledgments: This research project has received funding from the Italian National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.5, funded by the European Union—NextGenerationEU. Project title: Open Test 2.0- Project Code: CUP H23C2200037000.

Authorship Declaration

The order of authors reflects the primary roles and contributions of each individual to the development of the manuscript. All authors have contributed substantially to the work and have approved the final version of the article. Luca Famà contributed to writing—original draft, methodology, investigation, formal analysis, conceptualization, and writing—review and editing. Antonio Cimino contributed to writing—original draft, methodology, formal analysis, conceptualization, and writing—review and editing. Vincenzo Corvello contributed to validation, conceptualization, and supervision. All authors confirm that the above contributions accurately represent their involvement in the study.



REFERENCES

Ahsan, F., & Anwer, F. (2024). A systematic literature review on software security testing using metaheuristics. *Automated Software Engineering, 31*, 44. https://doi.org/10.1007/s10515-024-00433-0

Ajorloo, S., Jamarani, A., Kashfi, M., Haghi Kashani, M., & Najafizadeh, A. (2024). A systematic review of machine learning methods in software testing. *Applied Soft Computing, 162*, 111805. https://doi.org/10.1016/j.asoc.2024.111805

Aniche, M., & Gerosa, M. A. (2015). Does Test-Driven Development improve class design? A qualitative study on developers' perceptions. *Journal of the Brazilian Computer Society, 21*, 15. https://doi.org/10.1186/s13173-015-0034-z

Arsawan, I. W. E., Koval, V., Rajiani, I., Rustiarini, N. W., Supartha, W. G., & Suryantini, N. P. S. (2022). Leveraging knowledge sharing and innovation culture into SMEs' sustainable competitive advantage. *International Journal of Productivity and Performance Management.* https://doi.org/10.1108/IJPPM-04-2020-0192

Augusto, C. (2020). Efficient test execution in end-to-end testing: Resource optimization through smart resource characterization and orchestration. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings* (pp. 152–154). ACM. https://doi.org/10.1145/3377812.3382177

Baldassarre, M. T., Caivano, D., Fucci, D., Juristo, N., Romano, S., Scanniello, G., & Turhan, B. (2021). Studying Test-Driven Development and its retainment over a six-month time span. *Information and Software Technology, 176*, 110937. https://doi.org/10.1016/j.infsof.2021.110937

Beck, K. (2003). *Test-driven development: By example.* Addison Wesley. https://books.google.it/books?id=zNnPEAAAQBAJ

Beck, K., & Fowler, M. (2000). *Extreme programming explained: Embrace change.* Addison Wesley Longman. https://books.google.it/books?id=u13hVoYVZa8C

Büyükyumukoğlu, G., Ersoy, E., Özdemir, M. S., Bağriyani, S., & Karahoca, A. (2017). Challenges and lessons learned in test automation: Experiences from telecommunications industry. *CEUR Workshop Proceedings, 1980*, 78–88.

Dabić, M., Obradović Posinković, T., Vlačić, B., & Gonçalves, R. (2023). A configurational approach to new product development performance: The role of open innovation, digital transformation and absorptive capacity. *Technological Forecasting and Social Change, 194*, 122720. https://doi.org/10.1016/j.techfore.2023.122720

Davis, J., & Nathan, L. P. (2015). Value sensitive design: Applications, adaptations, and critiques. In J. van den Hoven et al. (Eds.), *Handbook of Ethics, Values, and Technological Design* (pp. 11–40). Springer.

Díaz, P., & Aedo, I. (2020). Combining software engineering and design thinking practices in the ideation process of augmented digital experiences. *Interacting with Computers, 32*(3), 279–295. https://doi.org/10.1093/iwc/iwaa020

Dima, A. M., & Maassen, M. A. (2018). From waterfall to agile software: Development models in the IT sector, 2006 to 2018. *Journal of International Studies, 11*(2). https://doi.org/10.14254/2071-8330.2018/11-2/21

Drobka, J., Noftz, D., & Raghu, R. (2004). Piloting XP on four mission-critical projects. *IEEE Software, 21*(6), 70–75. https://doi.org/10.1109/MS.2004.47

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology, 50*(9–10), 833–859. https://doi.org/10.1016/j.infsof.2008.01.006

Fraser, H. (2009). Designing the business: New models for success. *Design Management Review, 20*(2), 55–65. https://doi.org/10.1111/j.1948-7169.2009.00008.x

Friedman, B., & Hendry, D. G. (2019). *Value Sensitive Design: Shaping Technology with Moral Imagination.* MIT Press. https://books.google.it/books?id=8ZiWDwAAQBAJ

Friedman, B., Howe, D. C., & Felten, E. (2012). A value sensitive design investigation of privacy enhancing tools in web browsers. *Decision Support Systems, 54*(1), 424–433. https://doi.org/10.1016/j.dss.2012.06.003

Friedman, B., Kahn, P., & Borning, A. (2006). Value sensitive design and information systems. In *Human-Computer Interaction in Management Information Systems: Foundations* (pp. 348–372). M.E. Sharpe.

Fucci, D., Erdogmus, H., Turhan, B., Oivo, M., & Juristo, N. (2017). A dissection of the Test-Driven Development process: Does it really matter to test-first or to test-last? *IEEE Transactions on Software Engineering, 43*(7), 597–614. https://doi.org/10.1109/TSE.2016.2616877

Fucci, D., Turhan, B., Juristo, N., Dieste, O., Tosun Misirli, A., & Oivo, M. (2015). Towards an operationalization of Test-Driven Development skills: An industrial empirical study. *Information and Software Technology, 68*, 82–97. https://doi.org/10.1016/j.infsof.2015.08.006

Gerdes, A., & Frandsen, T. F. (2023). A systematic review of almost three decades of value sensitive design (VSD): What happened to the technical investigations? *Ethics and Information Technology, 25*, 26. https://doi.org/10.1007/s10676-023-09700-2

Manders-Huits, N. (2011). What values in design? The challenge of incorporating moral values into design. *Science and Engineering Ethics, 17*, 271–287. https://doi.org/10.1007/s11948-010-9198-2

Munir, H., Moayyed, M., & Petersen, K. (2014). Considering rigor and relevance when evaluating test-driven development: A systematic review. *Information and Software Technology, 56*(4), 375–394. https://www.sciencedirect.com/science/article/abs/pii/S0950584914000135

Parsa, S., Zakeri-Nasrabadi, M., & Turhan, B. (2025). Testability-driven development: An improvement to TDD efficiency. *Computer Standards & Interfaces, 91*, 103877. https://doi.org/10.1016/j.csi.2024.103877

Peters, D., Loke, L., & Ahmadpour, N. (2020). Toolkits, cards and games: A review of analogue tools for collaborative ideation. *CoDesign, 17*(4), 410–434. https://doi.org/10.1080/15710882.2020.1715444

Ribeiro-Navarrete, S., Botella-Carrubi, D., Palacios-Marqués, D., & Orero-Blat, M. (2021). The effect of digitalization on business performance: An applied study of KIBS. *Journal of Business Research, 126*, 319–326. https://doi.org/10.1016/j.jbusres.2020.12.065

Serrat, O. (2017). Design thinking. In *Knowledge Solutions*. Springer. https://doi.org/10.1007/978-981-10-0983-9_18

Sharma, R., & Singh, D. J. N. (2023). An intelligent human-centred software development framework. *International Journal of Intelligent Systems and Applications in Engineering.*

Siniaalto, M., & Abrahamsson, P. (2007). A comparative case study on the impact of Test-Driven Development on program design and test coverage. In *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)* (pp. 275–284). IEEE. https://doi.org/10.1109/ESEM.2007.12

Szabó, B., & Hercegfi, K. (2022). User-centered approaches in software development processes. *Software: Evolution and Process.* https://doi.org/10.1002/smr.2501

Tosun, A., Dieste, O., Fucci, D., et al. (2017). An industry experiment on the effects of Test-Driven Development on external quality and productivity. *Empirical Software Engineering, 22*, 2763–2805. https://doi.org/10.1007/s10664-016-9490-0

Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). Software testing with large language models: Survey, landscape, and vision. *IEEE Transactions on Software Engineering, 50*(4), 911–936. https://doi.org/10.1109/TSE.2024.3368208

Winkler, T., & Spiekermann, S. (2021). Twenty years of value sensitive design: A review of methodological practices in VSD projects. *Ethics and Information Technology, 23*, 17–21. https://doi.org/10.1007/s10676-018-9476-2

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in software engineering.* Springer.

Xu, H., Crossler, R. E., & Bélanger, F. (2012). A value sensitive design investigation of privacy enhancing tools in web browsers. *Decision Support Systems, 54*(1), 424–433. https://doi.org/10.1016/j.dss.2012.06.003

Young, M. (2008). *Software testing and analysis: Process, principles, and techniques.* John Wiley & Sons. https://ix.cs.uoregon.edu/~michal/book/Samples/book.pdf