DOI: 10.15514/ISPRAS-2024-36(6)-9

Thematic Synthesis of Behavior-Driven Development: An Analytical Approach

V. M. Arredondo-Reyes, ORCID: 0009-0002-0218-8177 <vctmanuel.arrys@gmail.com>
S. Domínguez-Isidro, ORCID: 0000-0002-9546-8233 <sauldominguez@uv.mx>
Á. J. Sánchez-García, ORCID: 0000-0002-2917-2960 <angesanchez@uv.mx>
J. O. Ocharán-Hernández, ORCID: 0000-0002-2598-1445 <jocharan@uv.mx>
Faculty of Statistics and Informatics, Veracruz University, Xalapa, México.

Abstract. Behavior-driven development (BDD) focuses on specifying system behavior through examples, fostering collaboration, and aligning development with business needs. This research provides a thematic synthesis of BDD, highlighting its challenges, benefits, and implications in software development. By analyzing 23 studies across four academic databases, the study identifies trends and themes in BDD adoption and implementation. The findings emphasize BDD's role in bridging the gap between technical and non-technical stakeholders, aligning software development with business goals. Despite initial adoption challenges, the study reveals significant long-term benefits in software quality and stakeholder satisfaction. Future research should focus on developing efficient training and tools to support BDD adoption in diverse environments.

Keywords: behavior-driven development; software development; BDD adoption; BDD implementation; BDD challenges; BDD benefits; thematic synthesis.

For citation: Arredondo-Reyes V. M., Domínguez-Isidro S., Sánchez-García Á. J., Ocharán-Hernández J. O. Thematic synthesis of behavior-driven development: an analytical approach. Trudy ISP RAN/Proc. ISP RAS, vol. 36, issue 6, 2024. pp. 161-178. DOI: 10.15514/ISPRAS-2024-3(6)-9.

Тематический синтез разработки, ориентированной на поведение: аналитический подход

В. М. Аррендондо-Рейес, ORCID: 0009-0002-0218-8177 <vctmanuel.arrys@gmail.com>
С. Домингес-Исидро, ORCID: 0000-0002-9546-8233 <sauldominguez@uv.mx>
А. Х. Санчес-Гарсия, ORCID: 0000-0002-2917-2960 <angesanchez@uv.mx>
Х. О. Очаран-Эрнандес, ORCID: 0000-0002-2598-1445 <jocharan@uv.mx>

Университет Веракруса, факультет статистики и информатики, Халапа, Мексика.

Аннотация. Разработка на основе поведения (BDD) фокусируется на определении поведения системы с помощью примеров, поощрений совместной работы и согласований разработки с потребностями бизнеса. В предлагаемой статье авторы описывают результаты изучения тематического синтеза BDD, подчеркивая его проблемы, преимущества и последствия для разработки программного обеспечения. Анализируя 23 исследования, ход которых отражен в четырех академических базах данных, исследование выявляет тенденции и направления в следовании принципам и реализации BDD. Авторами подчеркивается роль BDD в преодолении разрыва между техническими и нетехническими заинтересованными сторонами, согласовании разработки программного обеспечения с бизнес-целями. Несмотря на первоначальные проблемы с внедрением BDD, проведенное исследование показывает его значительное долгосрочное и благотворное влияние на качество программного обеспечения, а также на достижение удовлетворенности заинтересованных сторон. Будущие исследования должны быть сосредоточены на разработке эффективного обучения и инструментов для поддержки внедрения BDD в различных средах.

Ключевые слова: разработка, основанная на поведении (BDD); разработка программного обеспечения; внедрение BDD; реализация BDD; проблемы BDD; преимущества BDD; тематический синтез.

Для цитирования: Аррендондо-Рейес В. М., Домингес-Исидро С., Санчес-Гарсия А. Х., Очаран-Эрнандес Х. О. Тематический синтез разработки, ориентированной на поведение: аналитический подход. Труды ИСП РАН, том 36, вып. 6, 2024 г., стр. 161–178 (на английском языке). DOI: 10.15514/ISPRAS–2024–36(6)–9.

1. Introduction

This paper builds on the research presented at the 2023 11th International Conference on Software Engineering Research and Innovation (CONISOFT) [1] by conducting a Systematic Literature Review (SLR) to analyze the challenges and benefits of Behavior-Driven Development (BDD). BDD, an evolution of Test-Driven Development (TDD), is a significant advancement in software development, focusing on defining and developing software based on system behavior rather than solely verifying functionality through tests [2]. This characteristic enables software development teams to focus on identifying, understanding, and subsequently building valuable features that interest businesses, ensuring they are implemented effectively [3].

BDD, with its proactive and collaborative approach, has significantly impacted the industry in recent years as teams strive to deliver high- quality software that meets stakeholder requirements [4]. By collaboratively defining expected system behavior at the outset, BDD allows early identification of potential issues, preventing costly misunderstandings and rework [5]. Furthermore, BDD promotes the development of automated tests that verify software behavior, aiding in the detection of regressions and ensuring that new features do not compromise existing functionality.

Since BDD continues to be adopted across the industry, understanding its benefits and challenges is crucial for successful implementation. In this context, we extend our previous analysis [1] in order to contribute to this understanding by offering a thematic synthesis of BDD's application, highlighting critical factors for its effective adoption, and providing recommendations to address common challenges. This thematic synthesis identifies trends and common themes across the studies analyzed, offering a documented overview of BDD's implementation.

The rest of the document is organized as follows: Section 2 reviews related work; Section 3 describes the materials and research methods; Section 4 summarizes the systematic review results. Section 5 discusses thematic synthesis. Section 6 addresses validity threats, and finally, Section 7 presents conclusions and future work.

2. Related work

The software industry has significantly evolved, integrating automatic data processing through ICTs into various social niches. As software demand grows, maintaining high- quality products is crucial, prompting the adoption of methods for better design, implementation, and maintenance of software systems [6]. This need has driven studies exploring the implications of these methods.

Three relevant studies were identified. Abushama et al. [7] systematically reviewed the impact of TDD and BDD on project success factors such as cost, time, and customer satisfaction, analyzing 31 studies. Their findings suggest that while BDD may incur higher costs and time, it tends to achieve greater customer satisfaction. Arnyndiasari et al. [8] reviewed Agile methodologies, including BDD, highlighting the benefits of integrating these practices to enhance development success. Farooq et al. [9] focused on BDD, emphasizing its role in clarifying requirements and bridging communication gaps between stakeholders and developers, leading to higher customer satisfaction. However, they noted potential challenges in BDD implementation.

Our study differs by analyzing software project environments where BDD has been implemented and identifying critical insights into its adoption, challenges, and utility.

Table 1. Comparison of related works.

Characteristic	Related works		
	[7]	[8]	[9]
Year	2020	2022	2023
Approach	Analysis of the impact of TDD and BDD on time, cost, and customer satisfaction.	Review Agile methodologies (TDD, BDD, DDD, MDD) and their effectiveness.	Evaluation of BDD and its impact on software development and product quality.
Research Questions	Impact of TDD and BDD on project success factors.	Characteristics and effects of TDD, BDD, DDD, and MDD.	Techniques to reduce ambiguities and communication gaps in BDD.
Findings	BDD achieves higher customer satisfaction compared to TDD; more research is necessary.	Integrating Agile methodologies can improve software development success.	BDD is effective in clarifying requirements and improving communication between stakeholders and developers.
Coverage	Systematic literature review (1999-2020, 31 studies).	Systematic literature review (2000-2021, 16 studies)	Systematic literature review (2010-2022, 31 studies) Development of framework and taxonomy for BDD.

3. Materials and method

This research followed the Systematic Literature Review (SLR) guidelines by Kitchenham et al. [10]. The method consists of three main stages: 1) planning, 2) Identifying the state-of-the-art in BDD, and 3) Interpreting the results. For the last stage, the method provided by Popay et al. [11] for narrative synthesis for the SLR was applied. The development of thematic synthesis was based on the process of Cruzes and Dybå [12]. Finally, to conduct the thematic synthesis, we based on the guide by Uştuk [13] on a thematic synthesis with MAXQDA.

3.1 Research Planning

The state-of-the-art analysis is guided by five research questions (RQs) aimed at i) Identifying and describing the characteristics of projects where BDD has been successfully implemented. ii) Determining the scenarios where BDD benefits software development. iii) Identifying challenges related to BDD implementation and how they can be addressed. iv) Specifying the knowledge needed to enhance BDD's adaptability, facilitating its adoption across different environments and teams. v) Documenting the advantages BDD brings to software projects, such as improved product quality and stakeholder satisfaction.

1. **RQ1.-** What are the characteristics of the projects in which BDD has been used to develop software?
2. **RQ2.-** What are the specific scenarios in which BDD benefits software development?
3. **RQ3.-** What are the challenges in the use of BDD?
4. **RQ4.-** What information should be known to increase the degree of adaptability of BDD?
5. **RQ5.-** What are the reported benefits of using BDD?

Concerning the search strategy, a search string was generated through an elicitation process presented in [10], resulting in the following:

("behavior driven development" OR "behavior-driven development" OR "behavioural-driven development") AND (tendencies OR benefits OR advantage OR trends)

Information sources for the automated search include four key databases: IEEE Xplore, ACM Digital Library, SpringerLink, and Science Direct, which store relevant proceedings and journal papers in software engineering and related fields.

The study selection process is divided into five phases, applying inclusion (IC) and exclusion criteria (EC). Phase 1 includes studies published between 2015 and 2023 (IC1) and written in English (IC2). Studies before 2015, book chapters, monographs, theses (EC1), and secondary studies (EC2) are excluded. Phase 2 includes papers with relevant search terms in their title, abstract, or keywords (IC3) and excludes demos or inaccessible works (EC3). In Phase 3, studies unrelated to software development (EC4) are excluded, while those with abstracts related to research questions (IC4), documented results (IC5), and published in selected sources (IC6) are included. Phase 4 excludes duplicates (EC5). Finally, Phase 5 includes works that directly answer a research question (IC7). This selection process is applied to both automated search and snowballing.

In order to measure the relevance and impact of the selected study on this research, we applied a seven-question checklist based on the criteria shown by Dybå and Dingsøyr [14].

1. **Q1:** Is the document based on research, or is it a "lessons learned" report based on an expert opinion?
2. **Q2:** Is there an explicit statement of research objectives?
3. **Q3:** Is there a sufficient description of the context in which the proposed methodology was tested?
4. **Q4:** Does the research design address the objectives adequately?
5. **Q5:** Was information obtained that addressed characteristics of the project in which the methodology was used?
6. **Q6:** Does the study provide value in research or practice?
7. **Q7:** Was the proposal for using the methodology evaluated?

The scores assigned to the papers reflect the quality of their contributions to this research. Papers are ranked based on their final score, ranging from 7 to 5. A score of 7 or higher is considered high-ranked, a score between 4 and less than six is considered average, and any score below four is deemed low-ranked.

4. Results

This section provides a summary of the results from the systematic literature review (SLR) presented in [1], as well as an overview of the thematic synthesis derived from the analysis

4.1 Selection of Primary Studies

We got 371 publications from the selected databases using the proposed search string. Twenty-five articles were selected after applying the inclusion and exclusion criteria to the databases, see Fig. 1.

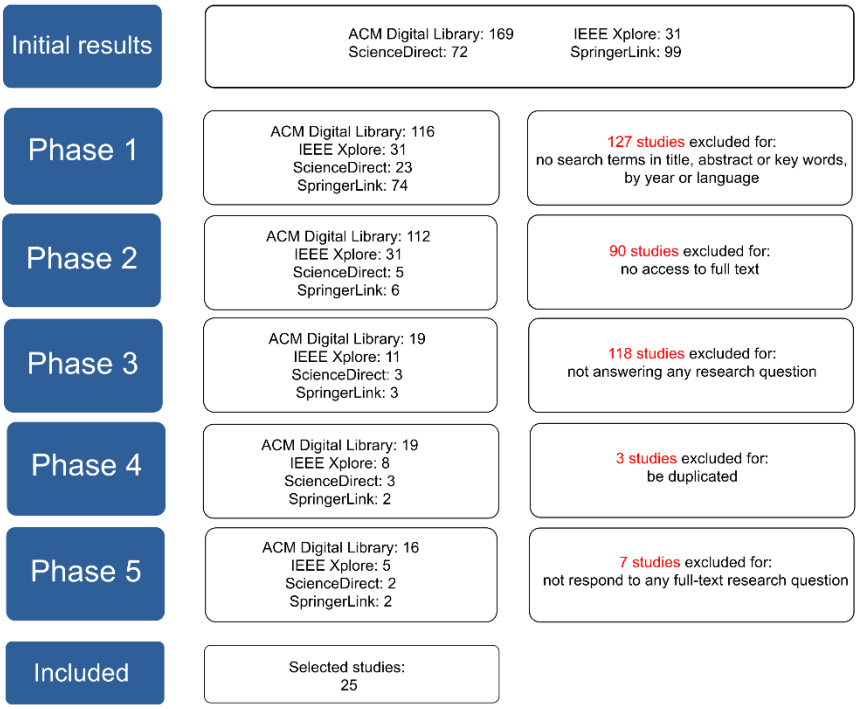

Fig. 1. Selection phases for automatic search.

Wohlin et al. [15] described the snowballing technique, which expanded the study pool by adding 550 backward and 158 forward studies. After applying the selection criteria, only three additional studies were included, resulting in 28 primary studies. Table 2 lists these selected studies and their quality scores based on questions Q1–Q6.

4.2 Studies Distribution

Behavior-driven development (BDD) research has increased significantly since 2018, with 54% of the work concentrated between 2018 and 2020 and 35% in the last three years. In the last two years, studies increased by 60% (see Fig. 2).

The International Conference on Software Engineering (ICSE) is the primary venue for publishing articles on BDD, accounting for 17% of the selected studies. ICSE is renowned for its comprehensive coverage of software engineering topics and attracts leading experts, fostering collaboration and knowledge exchange. The International Conference Proceedings Series (ICPS) follows, accounting for 13% of the selected studies. ICPS is recognized for its interdisciplinary approach, making it an attractive venue for BDD research due to its encouraging cross-pollination

of ideas and perspectives. The Brazilian Symposium on Software Engineering also emerges as a significant venue, hosting 8% of the selected studies. Although not as widely recognized as ICSE or ICPS, it provides a valuable platform for regional researchers to contribute to the discourse on BDD.

Table 2. Studies selected ordered by quality.

Reference	Year	Data base	Quality score
[16]	2021	Science Direct	7
[17]	2018	ACM Digital Library	7
[18]	2018	IEEE Xplore	7
[19]	2023	SpringerLink	7
[20]	2018	ACM Digital Library	7
[21]	2019	ACM Digital Library	6
[22]	2019	ACM Digital Library	6
[23]	2020	ACM Digital Library	6
[24]	2020	ACM Digital Library	6
[25]	2016	SpringerLink	6
[26]	2023	IEEE Xplore	6
[27]	2023	ACM Digital Library	6
[28]	2015	IEEE Xplore	5
[29]	2022	Science Direct	5
[30]	2018	ACM Digital Library	5
[31]	2020	ACM Digital Library	5
[32]	2020	IEEE Xplore	5
[33]	2021	ACM Digital Library	5
[34]	2022	IEEE Xplore	5
[35]	2020	SpringerLink	5
[36]	2020	ACM Digital Library	5
[37]	2019	ACM Digital Library	5
[38]	2021	IEEE Xplore	5
[39]	2023	ACM Digital Library	5
[40]	2023	ACM Digital Library	5
[41]	2018	IEEE Xplore	4
[42]	2017	ACM Digital Library	4
[5]	2018	ACM Digital Library	3

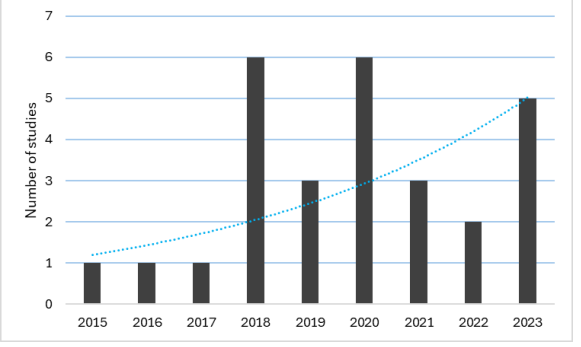

Fig. 2. Selected studies by year. The dashed line indicates an upward trend.

4.3 Thematic Synthesis

The thematic synthesis followed the 21-step approach by Cruzes and Dybå [12]. The process began with information understanding, where the 23 selected studies were thoroughly reviewed to identify relevant text segments. These segments were labeled and coded, resulting in 51 initial codes. After peer review, the list was refined to 27 codes, which were then translated into five cohesive themes.

1. **Development Aspects**: Explores project characteristics influencing BDD implementation, addressing RQ1 and RQ2.
2. **Benefits**: Focuses on BDD's positive impacts on the development cycle, addressing RQ5.
3. **Best Practices**: Delves into effective BDD implementation practices aligned with RQ4.
4. **Difficulties**: Examines challenges in BDD usage corresponding to RQ3.
5. **Usage Recommendations**: Offers expert guidance for BDD implementation, also addressing RQ4.

The thematic map (Fig. 3) illustrates the hierarchical organization of these themes and subthemes, effectively answering the research questions. It categorizes findings into key domains such as Development Aspects, Benefits, Best Practices, Recommendations, and Challenges.

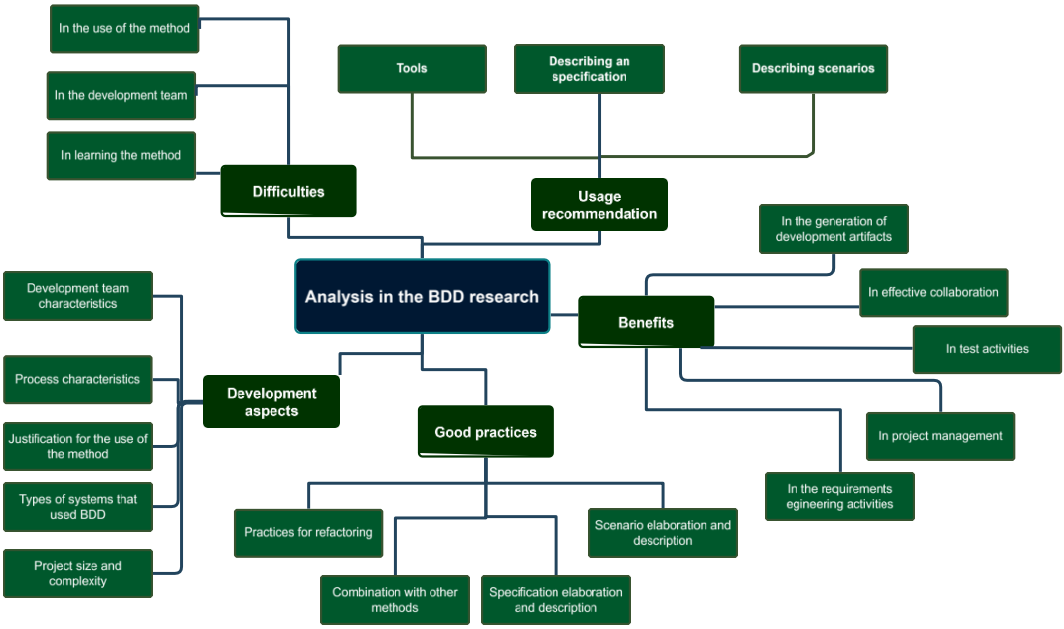

Fig. 3. Thematic map proposed.

5. Discussion and analysis

Below are the five main BDD themes identified in our research.

5.1 Development Aspects

The use of BDD in software projects encompasses various characteristics, from team details to process features, the justification for its adoption, project size, and the types of systems developed. These aspects are explored in detail below.

5.1.1 Development Team Characteristics

Successful BDD implementation depends on team characteristics. In large-scale and geographically distributed teams, BDD enhances communication and coordination [16, 22]. In educational settings, writing acceptance tests before development integrates quality early [17, 21]. The team's prior experience also affects BDD's efficiency and productivity [21, 28].

5.1.2 Process Characteristics

BDD's application varies by context. In distributed systems, it emphasizes reusing scenarios and test steps [16, 29, 30]. In agile projects, BDD improves quality, organization, and collaboration [23, 31]. During requirements validation, BDD enhances organization and collaboration throughout development [23].

5.1.3 Justification for Adopting BDD

BDD is adopted in large-scale projects to address challenges like team coordination and requirements management. It facilitates requirement documentation and coordination in telecommunications and enterprise systems [16, 24, 29]. In geographically distributed projects, BDD mitigates communication barriers and fosters a common language [5, 22]. For critical systems, such as in the automotive industry, BDD improves requirement specification and system validation [24, 34].

5.1.4 Project Size and Complexity.

BDD’s effectiveness varies with project size and complexity. While initially suited for medium-sized projects, it also benefits larger projects [32]. However, extensive BDD test suites can increase maintenance and comprehension complexity [35].

5.1.5 Types of Systems Using BDD

BDD is applied across various sectors and technologies. It is used in complex projects like telecommunications and microservices architectures [16, 29]. In critical systems, such as automotive, BDD ensures system integrity and reliability [24, 28, 34]. It also adapts to emerging technologies and diverse development paradigms [21, 32, 42].

5.2 Benefits

This subsection covers how BDD enhances various aspects of the development cycle, including requirements management and the creation of valuable artifacts. It also improves quality, stakeholder communication, and critical phases like requirements and testing.

5.2.1 Effective Collaboration

BDD enhances collaboration and communication between quality engineers and business analysts. Scenario refactoring improves interaction, and clear visualization of test scenarios aids comprehension even for non-technical stakeholders [29, 32]. Precise language in behavior scenarios reduces misunderstandings, fosters better project management, and improves communication in geographically distributed teams [5, 36, 41]. BDD also integrates clients more closely through scenario-based documentation, enhancing collaboration and product quality [18, 24, 28].

5.2.2 Artifact Generation

BDD improves artifact generation by expressing requirements as executable test cases, reducing rework and saving time [16, 21]. It provides automated acceptance tests, enhances requirement elicitation, and creates “living documentation” that evolves with the system [18, 23, 24, 25, 28]. The approach also facilitates test case reusability, benefiting development and verification [16, 18].

5.2.3 Testing Activities

BDD simplifies and structures testing activities, improving efficiency and automation. It organizes tests by features and scenarios, aids maintenance, and enhances test completeness and stability [17, 18, 22]. BDD improves predictability and confidence in code stability by defining behaviors before implementation and automating tests [18, 31, 41]. Overall, BDD enhances test efficiency, completeness, and process ease [5, 17, 19, 20, 33].

5.2.4 Requirement Engineering Activities

BDD enhances the quality and understanding of requirements by expressing them as executable test cases. The Given-When- Then format clarifies business perspectives and improves requirement clarity [16, 17]. BDD facilitates discussions, improves traceability between requirements and code, and creates "living documentation" [5, 23, 33]. It also addresses security requirements and reduces ambiguities [16, 17, 28, 42].

5.2.5 Project Management

BDD supports project management by improving scenario grouping reducing development time and costs. Early scenario development enhances efficiency, and "living documentation" provides continuous updates [16, 18]. BDD improves code quality and productivity, benefiting exploratory testing and product quality [18, 41].

5.3 Best Practices

This section provides practical guidelines for applying BDD, covering maintainable specifications, new specification creation, and scenario refactoring techniques.

5.3.1 Refactoring Practices

Refactoring is essential for improving software quality and maintainability. Key practices include [29]:

Identification of Areas for Refactoring

- **Preprocessing:** Store each BDD specification in a separate file with the name on the first line and steps on subsequent lines. Remove BDD keywords for easier comparison.
- **Measurement:** Using automated scripts, calculate normalized compression similarity (NCS) and Similarity Ratio (SR) for all specification pairs.
- **Ranking:** Analyze and rank the NCS and SR values to determine a similarity between specifications.

Careful Application of Refactoring

- **Merging:** Combine specifications with common lines and minimal differences.
- **Restructuring:** Create new statements from common statements with different specifications.
- **Deleting:** Remove duplicate or functionally identical specifications.
- **Renaming:** Rename specifications with similar names but different functionalities to avoid ambiguity.

Validation to Preserve Behavior

- Ensure that refactoring does not alter the software's behavior.

5.3.2 Combination with Other Methods

Combining BDD with other methods can enhance communication, address complexity, and reduce gaps in distributed teams. Benefits include improved security verification, communication, and reduced inconsistencies in early development phases [17, 28, 32, 42]. Various studies suggest integrating BDD with other techniques to improve development quality.

5.3.3 Elaboration and Description of Specifications

Structural Practices

- Develop system-level feature files and hooks for effective integration testing [16, 35].

Practices for Developing New BDD Specifications

- **Specify New Behaviors:** Product managers should write new behaviors based on customer requests.
- **Develop System-Level Feature Files:** Create detailed, executable feature files that outline approved behaviors [16].

Practices Related to SBVR and Event-B

- **Determine Business Objectives:** Collaborate with clients and analysts to establish project objectives [42].
- **Define Software Functionalities:** Refine goals into a list of features with a specific format [42].
- **Define Acceptance Criteria:** Create scenarios representing acceptance criteria using the given-when-then format [29].

Improvement Areas

- Regular feedback is crucial for early correction and alignment with team objectives [16].

5.3.4. Elaboration and Description of Scenarios

We identified four principal pieces of information related to writing scenarios BDD: formal redaction, simplifying scenarios, evading ambiguity, and establishing a limit for the step in a scenario to adequate comprehension.

Key Information for Writing BDD Scenarios

- **Abstraction Level:** Maintain an appropriate level of detail to balance understanding and code complexity [35].
- **Reuse of Step Phrases:** Use existing steps to enhance readability and maintainability [32, 39].
- **Balance Generic and Specific Steps:** Combine generic steps with parameters and specific steps to improve readability and execution. Reusing steps with parameters and generic names like "When the user clicks on the '<element name>' element on the '<page name>' page" is helpful. [32, 39].
- **Limit Actions in Scenarios:** Each scenario can have only a single "When" action. Split scenarios with multiple actions or move extra actions to the "Given" section [32, 39].
- **Indent "And" Steps:** Use "And" steps for improved readability [32].
- **Seek Reusable Behaviors:** Avoid redundant development and testing of similar behaviors [16].
- **Address Duplication:** Automate duplicate searches, refactor code frequently and adhere to the Single Responsibility Principle (SRP) [18].

Scenario Structuring Recommendations

- **Naming Pattern:** Use the "When..., then..." pattern for concise scenario descriptions [26].
- **Step Count:** Limit steps per scenario to 12 for better readability [26].
- **Step Order:** Follow the "Given", "When", and "Then" order. Multiple "When-Then" combinations may indicate the need for separate scenarios [26].
- **Perspective:** Write scenarios from a third-person perspective to avoid ambiguity, e.g., "When the user clicks on the button" instead of "When I click on the button" [26].
- **Domain Vocabulary:** Use precise terms and avoid duplicates for clarity. Minimize technical jargon to ensure all stakeholders understand [35].

5.4 Difficulties

Challenges in adopting the BDD method can be categorized into difficulties related to learning the method, development team issues, and the practical use of BDD. These challenges are further detailed below:

5.4.1 Learning the Method

The high learning curve associated with BDD presents significant obstacles. Resistance to BDD may arise from a lack of testing culture, as BDD requires a shift in perspective. The process's lack of visual appeal can also hinder adoption [24]. The steep learning curve is often exacerbated by limited experience with BDD, leading to initial difficulties and resistance, particularly in teams unfamiliar with the method [31]. These issues highlight the need for a supportive culture and thorough training to ease the transition to BDD [16, 29, 41].

5.4.2. Development Team Challenges

The lack of experience and commitment within the development team significantly impacts the successful adoption of BDD. Inexperienced team members may struggle with proper scenario specification, leading to issues such as scenario duplication and incomplete scenarios [18]. Communication and collaboration are also hindered by a lack of commitment, which is crucial for the success of agile methodologies [41]. The absence of method knowledge among team members further exacerbates these challenges, making it difficult to effectively implement and maintain BDD practices [23, 31].

5.4.3. Using the Method

Implementing BDD presents several practical challenges:

- **Scenario Management:** Managing scenarios in large-scale environments is complex due to the dynamic nature of requirements and the need for frequent iterations with domain experts. Maintaining an accurate record of behavior changes across multiple stakeholders adds to the complexity [16].
- **Adapting to New Environments or Requirements:** Modifying BDD specifications to reflect new business policies or environments can be challenging, particularly in large-scale projects. Updating BDD frameworks or adapting them to new requirements may result in duplicated efforts and slow down development [18].
- **Adopting BDD Tools and Technologies:** Introducing new BDD tools in large projects requires significant time and effort. Training is essential for achieving productivity in a BDD environment. Additionally, challenges arise when updating BDD frameworks across multiple microservices, requiring careful evaluation of tool suitability [23, 28, 41].

- **Comprehending BDD Specifications:** Understanding BDD specifications can be difficult, particularly when duplication in specifications hinders comprehension and unnecessarily prolongs test suite execution [18].
- **Specifying Scenarios:** Scenario specification in BDD is complex, especially in large-scale projects where requirements evolve over time. The lack of initial clarity and the need for frequent iterations complicate scenario specification, making it a challenging task [16, 23, 30, 31].
- **Specification Size:** Large projects pose additional challenges due to the exponential growth of possible scenarios. The complexity of managing and maintaining these scenarios can be overwhelming [36].
- **Maintainability of BDD Specifications:** The maintenance of BDD specifications is particularly challenging in large-scale projects. The high cost and complexity of maintaining these specifications can deter teams from adopting automated acceptance testing [29]. Effective maintenance strategies, such as refactoring, are necessary to manage the growing complexity and ensure the long-term success of BDD [16, 18, 35].

5.5 Usage Recommendations

Implementing BDD goes beyond adopting tools and practices; it requires understanding the guidelines and best practices for specifying requirements and crafting scenarios. This section provides key recommendations from experienced practitioners, divided into three areas: specification description, scenario elaboration, and tool usage.

5.5.1 For the Description of a Specification

The specification in BDD serves as a document that describes the desired system behaviors from a high-level perspective. It communicates how the software should meet requirements in natural language.

- **Limit actions per scenario**: Restricting the number of actions in each scenario maintains clarity and conciseness. This practice ensures that both technical and non-technical stakeholders can quickly grasp the system's functionality without unnecessary complexity [27, 32, 37].
- **Preserve domain vocabulary**: Using consistent domain-specific terms promotes shared understanding among teams, enhancing collaboration and ensuring alignment on the system's goals and requirements [35, 37].
- **Conserve a few steps**: Focus on essential steps to maintain clarity and conciseness in each scenario. This approach ensures scenarios remain understandable, especially for those not directly involved in development [32, 35].
- **Eliminate technical vocabulary**: Avoiding technical terms makes specifications accessible to all stakeholders, facilitating effective communication at the initial stage [35].

5.5.2. For the Description of Scenarios

Scenarios are concrete instances that exemplify how the system should behave in specific contexts.

- **Each scenario tests one thing**: Focus each scenario on testing a specific functionality or behavior, making it easier to identify issues during execution [35, 40].
- **Make descriptive titles**: Clear and descriptive titles help quickly identify the purpose of each test case [35].
- **Oriented towards customer benefit**: Write scenarios from the perspective of the benefit they offer to the end user, ensuring alignment with customer expectations [35].

- **Make an explicit and verifiable description**: Scenarios should be concise, clear, and easily verifiable, facilitating execution and ensuring understandable results [37].
- **Maintain singularity in scenarios**: Each scenario should clearly contribute to the overall quality assurance of the software, ensuring each test case adds value [37].
- **Avoid ambiguities**: Clarity is key; avoiding ambiguities ensures reliable test results with no room for misinterpretation [37].

5.5.3. Tools

Selecting and using tools in BDD is critical. Various studies highlight popular open-source tools such as Cucumber, Concordion, JBehave, FitNesse, and SpecFlow, recognized for their role in facilitating BDD processes [16, 29, 32, 41].

- **Obsolete Tools**: Some tools, including StoryQ, JDave, NBehave, Easyb, and BDDfy, are no longer actively maintained, underscoring the importance of choosing up- to-date tools with active community support [38].
- **Documentation Evaluation**: Clear and comprehensive documentation is vital for efficient adoption and learning, allowing teams to maximize tool capabilities [29, 38].
- **Consideration of Reference Projects**: Reviewing reference projects that use the selected tools can provide practical insights and improve BDD implementation [18].

Selecting tools for development is important; when using BDD, technical functionality, currency, documentation, and an active user community must be considered. Evaluating IDE plugins provides valuable information on how tools facilitate collaboration and behavior specification within the development environment.

6. Validity threats

We acknowledge potential threats to the validity of our results but have taken measures to mitigate them. One potential threat involves the study search and selection process, which relies on the researcher's judgment and includes non- English languages [43-44]. To address this, we adhered to guidelines by Kitchenham et al. [10]. Peer reviews were conducted by at least three authors, following the coding and theming process described by Cruzes and Dybå [12]. Additionally, we utilized the MAXQDA tool for thematic synthesis [13]. To ensure the relevance of selected studies, we employed the snowballing method [15], conducting one forward and one backward iteration. While limitations, such as excluding studies due to restricted access, are recognized, our findings offer a comprehensive understanding of BDD's applications, benefits, and challenges. Our aim is not to provide prescriptive guidance or solutions but to enlighten and inform.

7. Conclusion

This research delved into essential aspects of BDD, focusing on its principles, differences from other methodologies, and practical applications through a systematic literature review. Key conclusions include insights into BDD's applications, benefits, and challenges, as well as the identification of recommended practices and common difficulties. While the study provided valuable perspectives, it is important to acknowledge limitations, such as the reliance on existing studies and gray literature, highlighting the need for further investigation.

The systematic review revealed that BDD enhances communication, collaboration, and adaptability while minimizing requirements misunderstandings. It also identified trends in BDD’s application, including its benefits for collaboration, testing, requirements engineering, and project management.

Our research methodology involved a thorough systematic review, with a carefully tailored search strategy and quality assessments to ensure reliability. This comprehensive approach offers a robust foundation for understanding BDD's implementation, challenges, advantages, and best practices.

In conclusion, this study has significantly contributed to the understanding of BDD, underscoring its contemporary relevance and growing interest in the software development community. It offers valuable insights for those considering the adoption of behavior-driven agile methodologies, promoting the creation of well-designed, precisely adapted software solutions.

References

[1]. V. M. Arredondo-Reyes, S. Domínguez-Isidro, A. J. Sánchez-García, and J. O. Ocharán-Hernández, "Benefits and Challenges of the Behavior-Driven Development: A Systematic literature review," in 11th International Conference in Software Engineering Research and Innovation (CONISOFT 2023), IEEE, 2023, pp. 1–6.

[2]. M. M. Moe and J. C. Sanchez, "International Journal of Trend in Scientific Research and Development (IJTSRD) Comparative Study of Test-Driven Development (TDD), Behavior-Driven Development (BDD) and Acceptance Test-Driven Development (ATDD) the Creative Commons Attribution License (CC BY 4.0)." [Online]. Available: http://creativecommons.org/licenses/by/4.0

[3]. J. Ferguson and D. North, BDD IN ACTION Behavior-Driven Development for the whole software lifecycle. [Online]. Available: www.it-ebooks.info

[4]. C. Solís and X. Wang, "A study of the characteristics of behaviour driven development," in Proceedings - 37th EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA 2011, pp. 383–387. doi: 10.1109/SEAA.2011.76.

[5]. L. Pereira, H. Sharp, C. De Souza, G. Oliveira, S. Marczak, and R. Bastos, "Behavior-driven development benefits and challenges: Reports from an industrial study," in Proceedings of the 19th International Conference on Agile Software Development: Companion, Association for Computing Machinery, 2018, pp. 1–4. doi: 10.1145/3234152.3234167.

[6]. "What is software development? IBM." [Online]. Available: https://www.ibm.com/topics/software-development

[7]. H. M. Abushama, H. A. Alassam, and F. A. Elhaj, "The effect of Test-Driven Development and Behavior-Driven Development on Project Success Factors: A Systematic Literature Review Based Study," in Proceedings of: 2020 International Conference on Computer, Control, Electrical, and Electronics Engineering, ICCCEEE 2020, Institute of Electrical and Electronics Engineers Inc., Feb. 2021. doi: 10.1109/ICCCEEE49695.2021.9429593.

[8]. D. Arnyndiasari, R. Ferdiana, and P. I. Santosa, "Software Practices for Agile Developers: A Systematic Literature Review," in 2022 1st International Conference on Information System and Information Technology, ICISIT 2022, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 238–243. doi: 10.1109/ICISIT54091.2022.9872874.

[9]. M. S. Farooq, U. Omer, A. Ramzan, M. A. Rasheed, and Z. Atal, "Behavior Driven Development: A Systematic Literature Review," IEEE Access, vol. 11, pp. 88008–88024, 2023, doi: 10.1109/ACCESS.2023.3302356.

[10]. B. A. Kitchenham, D. Budgen, and P. Brereton, Evidence-Based Software Engineering and Systematic Reviews. Chapman & Hall/CRC, 2015.

[11]. J. Popay et al., "Guidance on the Conduct of Narrative Synthesis in Systematic Reviews," 2006.

[12]. D. S. Cruzes and T. Dybå, "Recommended Steps for Thematic Synthesis in Software Engineering," in 2011 International Symposium on Empirical Software Engineering and Measurement, IEEE, 2011, pp. 275-284. doi: 10.1109/ESEM.2011.36.

[13]. Ö. Uştuk, "Thematic Analysis with MAXQDA: Step- by-Step Guide," MAXQDA. [Online]. Available: https://www.maxqda.com/blogpost/thematic-analysis- with-maxqda-step-by-step-guide.

[14]. T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," Inf Softw Technol, vol. 50, no. 9, pp. 833–859, 2008, doi: 10.1016/j.infsof.2008.01.006.

[15]. C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, ACM, 2014, pp. 1–10. doi: 10.1145/2601248.2601268.

[16]. M. Irshad, R. Britto, and K. Petersen, "Adapting Behavior Driven Development (BDD) for large-scale software systems," Journal of Systems and Software, vol. 177, no. 110944, pp. 1–20, 2021, doi: 10.1016/j.jss.2021.110944.

[17]. H. Bünder and H. Kuchen, "Towards Behavior-Driven Graphical User Interface Testing," in Proceedings of the ACM Symposium on Applied Computing, Association for Computing Machinery, 2019, pp. 1742–1751. doi: 10.1145/3297280.3297450.

[18]. Y. Wang and S. Wagner, “Combining STPA and BDD for safety analysis and verification in agile development: A controlled experiment,” in Agile Processes in Software Engineering and Extreme Programming. XP 2018. Lecture Notes in Business Information Processing, Springer, Cham, 2018, pp. 37–53. doi: 10.1007/978-3-319-91602-6_3.

[19]. A. Scandaroli, R. Leite, A. H. Kiosia, and S. A. Coelho, “Behavior-Driven Development as an Approach to Improve Software Quality and Communication across Remote Business Stakeholders, Developers and QA: Two Case Studies,” in 2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE), 2019, pp. 105-110. doi: 10.1109/ICGSE.2019.00030.

[20]. M. Rahman and J. Gao, “A reusable automated acceptance testing architecture for microservices in behavior-driven development,” in 9th IEEE International Symposium on Service-Oriented System Engineering, IEEE SOSE 2015, Institute of Electrical and Electronics Engineers Inc., 2015, pp. 321–325. doi: 10.1109/SOSE.2015.55.

[21]. M. Irshad, J. Börstler, and K. Petersen, “Supporting refactoring of BDD specifications—An empirical study,” Inf Softw Technol, vol. 141, no. 106717, pp. 1–13, 2022, doi: 10.1016/j.infsof.2021.106717.

[22]. A. Dimanidis, K. C. Chatzidimitriou, and A. L. Symeonidis, “A Natural Language Driven Approach for Automated Web API Development: Gherkin2OAS,” in Companion Proceedings of the The Web Conference 2018, Association for Computing Machinery, 2018, pp. 1869–1874. doi: 10.1145/3184558.3191654.

[23]. N. Nascimento, A. R. Santos, A. Sales, and R. Chanin, “Behavior-Driven Development: An Expert Panel to evaluate benefits and challenges,” in Proceedings of the XXXIV Brazilian Symposium on Software Engineering, Association for Computing Machinery, 2020, pp. 41–46. doi: 10.1145/3422392.3422460.

[24]. N. Nascimento, A. R. Santos, A. Sales, and R. Chanin, “Behavior-Driven Development: A case study on its impacts on agile development teams,” in Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops, Association for Computing Machinery, 2020, pp. 109–116. doi: 10.1145/3387940.3391480.

[25]. R. K. Lenka, S. Kumar, and S. Mamgain, “Behavior Driven Development: Tools and Challenges,” in Proceedings - IEEE 2018 International Conference on Advances in Computing, Communication Control and Networking, ICACCCN 2018, Institute of Electrical and Electronics Engineers Inc., pp. 1032–1037. doi: 10.1109/ICACCCN.2018.8748595.

[26]. O. Bezsmertnyi, N. Golian, V. Golian, and I. Afanasieva, “Behavior Driven Development Approach in the Modern Quality Control Process,” in 2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020 - Proceedings, Institute of Electrical and Electronics Engineers Inc., 2020, pp. 217–220. doi: 10.1109/PICST51311.2020.9467891.

[27]. T. R. Silva and B. Fitzgerald, “Empirical findings on bdd story parsing to support consistency assurance between requirements and artifacts,” in Proceedings of the 25th International Conference on Evaluation and Assessment in Software Engineering, Association for Computing Machinery, 2021, pp. 266–271. doi: 10.1145/3463274.3463807.

[28]. C. Wiecher, S. Japs, L. Kaiser, J. Greenyer, R. Dumitrescu, and C. Wolff, “Scenarios in the loop: Integrated requirements analysis and automotive system validation,” in 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS-C 2020 - Companion Proceedings, Association for Computing Machinery, Inc, 2020, pp. 199–208. doi: 10.1145/3417990.3421264.

[29]. C. Lauer and C. Sippl, “Benefits of Behavior Driven Development in Scenario-based Verification of Automated Driving,” in 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), Institute of Electrical and Electronics Engineers (IEEE), 2022, pp. 105–110. doi: 10.1109/itsc55140.2022.9922498.

[30]. F. L. Siqueira, T. C. de Sousa, and P. S. Silva, “Using BDD and SBVR to Refine Business Goals into an Event-B Model: A Research Idea,” in 2017 IEEE/ACM 5th International FME Workshop on Formal Methods in Software Engineering (FormaliSE), 2017, pp. 31–36. doi: 10.1109/FormaliSE.2017.5.

[31]. T. R. Silva, J. L. Hak, and M. Winckler, “Testing Prototypes and Final User Interfaces Through an Ontological Perspective for Behavior-Driven Development,” in Bogdan, C., et al. Human-Centered and Error-Resilient Systems Development. HESSD HCSE 2016 2016. Lecture Notes in Computer Science, Springer, Cham, 2016, pp. 86–107. doi: 10.1007/978- 3-319-44902-9_7.

[32]. L. P. Binamungu, S. M. Embury, and N. Konstantinou, “Characterising the Quality of Behaviour Driven Development Specifications,” in Agile Processes in Software Engineering and Extreme Programming. XP 2020. Lecture Notes in Business Information Processing, Springer, Cham, 2020, pp. 87–102. doi: 10.1007/978-3-030-49392-9_6.

[33]. T. R. Silva, M. Winckler, and H. Trætteberg, "Ensuring the Consistency between User Requirements and Task Models: A Behavior-Based Automated Approach," Proc. ACM Hum.-Comput. Interact., vol. 4, no. EICS, pp. 1–32, 2020, doi: 10.1145/3394979.

[34]. L. P. Binamungu, S. M. Embury, and N. Konstantinou, "Maintaining behaviour driven development specifications: Challenges and opportunities," in 25th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2018 - Proceedings, Institute of Electrical and Electronics Engineers Inc., 2018, pp. 175–184. doi: 10.1109/SANER.2018.8330207.

[35]. T. N. Kudo, R. de F. Bulcão-Neto, V. V. G. Neto, and A. M. R. Vincenzi, "Aligning requirements and testing through metamodeling and patterns: design and evaluation," Requir Eng, vol. 28, no. 1, pp. 97–115, Mar. 2023, doi: 10.1007/s00766-022-00377-5.

[36]. Y. Wang, D. R. Degutis, and S. Wagner, "Speed up BDD for safety verification in agile development: A partially replicated controlled experiment," in ACM International Conference Proceeding Series, Association for Computing Machinery, 2018. doi: 10.1145/3234152.3234181.

[37]. G. Oliveira, S. Marczak, and C. Moralles, "How to evaluate BDD scenarios' quality?" in Proceedings of the XXXIII Brazilian Symposium on Software Engineering, Association for Computing Machinery, 2019, pp. 481–490. doi: 10.1145/3350768.3351301.

[38]. N. Patkar, A. Chis, N. Stulova, and O. Nierstrasz, "Interactive Behavior-driven Development: A Low-code Perspective," in 24th International Conference on Model-Driven Engineering Languages and Systems, MODELS-C 2021, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 128–137. doi: 10.1109/MODELS-C53483.2021.00024.

[39]. T. Zameni, P. Van Den Bos, J. Tretmans, J. Foederer, and A. Rensink, "From BDD Scenarios to Test Case Generation," in Proceedings - 2023 IEEE 16th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2023, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 36–44. doi: 10.1109/ICSTW58534.2023.00019.

[40]. S. Rodriguez, J. Thangarajah, M. Winikoff, and M. 2023 Winikoff, "A Behaviour-Driven Approach for Testing Requirements via User and System Stories in Agent Systems. In Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems," 2023.

[41]. S. Heng, K. Tsilionis, and Y. Wautelet, "Building User Stories and Behavior Driven Development Scenarios with a Strict Set of Concepts: Ontology, Benefits and Primary Validation," in Proceedings of the ACM Symposium on Applied Computing, Association for Computing Machinery, Mar. 2023, pp. 1422–1429. doi: 10.1145/3555776.3577696.

[42]. J. Saraiva and S. Soares, "Adoption of the LGPD Inventory in the User Stories and BDD Scenarios Creation," in ACM International Conference Proceeding Series, Association for Computing Machinery, Sep. 2023, pp. 416–421. doi: 10.1145/3613372.3613375.

[43]. J. A. Valdivia, A. Lora-González, X. Limón, K. Cortes-Verdin, and J. O. Ocharán-Hernández, "Patterns Related to Microservice Architecture: A Multivocal Literature Review," Programming and Computer Software, vol. 46, no. 8, pp. 594-608, Dec. 2020, doi: 10.1134/S0361768820080253.

[44]. R. Moguel-Sánchez, C. S. Sergio Martínez-Palacios, J., O. Ocharán-Hernández, X. Limón, and A. J. Sánchez-García, "Bots in Software Development: A Systematic Literature Review and Thematic Analysis," Programming and Computer Software, vol. 49, no. 8, pp. 712-734, Dec. 2023, doi: 10.1134/S0361768823080145.

Информация об авторах / Information about authors

Виктор Мануэль АРРЕДОНДО-РЕЙЕС – студент бакалавриата по направлению программной инженерии факультета статистики и информатики Университета Веракруса (Мексика). Сфера научных интересов: проектирование, разработка и тестирование программного обеспечения, инженерия требований к программному обеспечению.

Víctor Manuel ARREDONDO-REYES, Undergraduate Student in Software Engineering, Facultad de Estadística e Informática, Universidad Veracruzana, Mexico (School of Statistics and Informatics, University of Veracruz). Research interests: software design, requirements engineering, software development, and software testing.

Саул ДОМИНГЕС-ИСИДРО имеет степень PhD по искусственному интеллекту, доцент факультета статистики и информатики Университета Веракруса (Мексика). Сфера научных

интересов: распределенные системы, разработка программного обеспечения, вычислительный интеллект, машинное обучение.

Saúl DOMÍNGUEZ-ISIDRO – PhD in artificial intelligence, associate professor at Facultad de Estadística e Informática, Universidad Veracruzana, Mexico (School of Statistics and Informatics, University of Veracruz). Research interests: distributed systems, software development, computational intelligence, and machine learning.

Ангел Хуан САНЧЕС-ГАРСИЯ имеет степень PhD по искусственному интеллекту, доцент факультета статистики и информатики Университета Веракруса (Мексика). Сфера научных интересов: программометрия, машинное обучение, прогнозирование затрат, эволюционные вычисления.

Ángel Juan SÁNCHEZ-GARCÍA – PhD in artificial intelligence, associate professor at Facultad de Estadística e Informática, Universidad Veracruzana, Mexico (School of Statistics and Informatics, University of Veracruz). Research interests: software measurement, machine learning, effort prediction, and evolutionary computation.

Хорхе Октавио ОЧАРАН-ЭРНАНДЕС имеет степень PhD по искусственному интеллекту, доцент факультета статистики и информатики Университета Веракруса (Мексика). Сфера научных интересов: архитектура программного обеспечения, инженерия требований к программному обеспечению, программная инженерия, проектирование прикладных интерфейсов.

Jorge Octavio OCHARÁN-HERNÁNDEZ – PhD in Computer Science, Associate Professor at the Facultad de Estadística e Informática, Universidad Veracruzana, Mexico (School of Statistics and Informatics, University of Veracruz). Research interests: software architecture, requirements engineering, software engineering, and API design.