Using Behaviour-driven Requirements Engineering for Establishing and Managing Agile Product Lines

An Observational Study

Heba Elshandidy[1], Sherif Mazen[2], Ehab Hassanein[3], Eman Nasr[4]
Information Systems Department
Faculty of Computers and AI, Cairo University, Cairo, Egypt[1, 2, 3]
Independent Researcher, Cairo, Egypt[4]

***Abstract*—Requirements engineering in agile product line engineering refers to both common and variability components establishing a software. Although it is conventional for the requirements engineering to take place in a dedicated upfront domain analysis phase, agile-based environments denounce such a proactive behaviour. This paper provides an observational study examining a reactive incremental requirement engineering approach called behaviour-driven requirements engineering. The proposed approach uses behaviour-driven development to establish and maintain agile product lines. The findings of the study are very promising and suggest the following: the approach is easy to understand and quick to learn; the approach supports the constantly changing nature of software development; and using behaviour-driven requirements engineering produces reliable and coherent requirements. In practice, the observational study showed that using the proposed approach saved time for development team and customers, decreased costs, improved the software quality, and shortened the time-to-market.**

Keywords*—Agile product line engineering; behaviour-driven requirements engineering; observational study; requirements engineering

I. Introduction

Agile product line engineering (APLE) has been gaining a momentum throughout the past decade due to its faster delivery, lesser time-to-market, and more involvement for customers in every development cycle. APLE is the resulting approach of merging agile software development (ASD) and software product line engineering (SPLE); that term was formally coined at the first APLE'06 Workshop [1]. The purpose of APLE is to overcome the weaknesses of both paradigms (i.e., ASD and SPLE) while maximizing their benefits. A software product line (SPL) is a family of software products that share a common set of features (i.e., core assets) in addition to the unique features (i.e., variability) associated to each product in the family that satisfy the different needs of the customers [2]. Thus, it is intuitive to deduce that agile product lines (APLs) are SPLs that are either developed in an entirely ASD environments or in traditional environments that adopt some of the ASD practices. ASD, on the other hand, is a group of incremental and iterative software development methodologies that advocate quick clean software delivery and customers' involvement throughout the project lifetime [3]. The work in this paper focuses on behaviour-driven development (BDD) which is an ASD process that encourages the collaboration between the different stakeholders (i.e., customers, quality assurance, developers, etc.) of a software project [4].

According to the studies in [5,6], there are eleven factors that contribute to the success of a software project. While eight of those factors are related to requirements engineering (RE), ten of them are related to ASD. RE is the process of identifying, analysing, documenting, and managing user requirements [7,8]. The overlapping between the RE-related and the ASD-related project's success factors indicates that they share the same goals. Thus, it is most likely that having an agile-based requirements engineering process highly increases the possibility of having a successful software project.

Having realised the advantages of APLE as a development approach and the critical role of RE in a project's success, it is inquisitive to know whether it is feasible to achieve an incremental agile-based RE approach for APLs using BDD in a real-life empirical case study.

The rest of the paper is organised as follows: Section II explains BDD in further details while Section III briefs the reader about related work. Section IV summarises the proposed behaviour-driven requirements engineering (BDRE) approach. Section V presents the conducted observational study. Section VI discusses the results of the study. Finally, Section VII concludes the paper.

II. Background

BDD was created to overcome the shortcomings of test-driven development (TDD). In particular, the starting point of testing, when and what to test, how much to test, understanding why a test fails, the need to have naming conventions for tests, and knowing whether a specification is met or whether the code delivers a business value [4]. BDD combines the general methods and practices of TDD with concepts from domain-driven design and objected-oriented analysis and design [4]. This provides a shared process and a common understanding to all the involved stakeholders (i.e., developers, designers, etc.). Thus, helps them to successfully collaborate on software development with well-defined outputs. As a result, BDD is capable of delivering working and tested software in shorter time-to-market while better managing traceability between the different artefacts of the system [4].

BDD has six main characteristics [4,9]:

- Ubiquitous language: which is a common language that enables customers and development teams to communicate without ambiguity. That language contains all the terms that will be used to define the behaviour of the systems. Although the structure of such languages emerges from the business domain model, BDD has its own pre-defined domain-independent ubiquitous language.
- Iterative decomposition process: since it is often difficult for the development team to find a starting point through which they can collect the customers' requirements, BDD works in an iterative manner to resolve that issue. Although the customers themselves might not have a clear view of the requirements they need, they surely know the business values and the behaviour they expect from the software project. As a consequence, the analysis process in BDD starts with the identification of the expected behaviour of the system, based on the intended business outcomes, which is later decomposed into a set of features. Each feature is then realised by a set of user stories and each user story is further described through a set of scenarios. A scenario is a specific instance of a particular user story that describes an actual context and output for that user story.
- Plain text description with User Story and Scenario templates: features, user stories, and scenarios are represented in plain text predefined templates using the BDD ubiquitous language. For example, to write a story, the following template is used:

  [UserStoryTitle] (One line describing the story)

  As a [Role]

  I want a [Feature]

  So that [Benefit]

  To write a scenario, the following template is used:

  Scenario 1: [Scenario Title]

  Given [context]

  And [Some more contexts]

  When [Event occurs]

  Then [Outcome]

  And/But [Some more outcomes]

  While a user story describes an activity that is done by a user in a given role, the scenario describes how the system should behave when it is in a specific state for a specific feature and an event happens. Both user stories and scenarios are directly mapped to tests.
- Executable acceptance tests (EATs) with mapping rules: acceptance tests (ATs) in BDD is the satisfaction condition(s) that determines whether the behaviour of a particular feature is successfully achieved. BDD inherits the characteristic of executable testing from automated TDD, where ATs are regarded as automated specifications that verify the behaviour/interaction of the object rather than its state. Mapping rules provide a standardised way of mapping from scenarios to test codes, thus, facilitates managing traceability between the different artefacts of the system.
- Readable behaviour oriented specification code: BDD emphasises the importance of including the code in the system's documentation. Thus, the code should be readable and the specifications should be part of the code. The mapping rules help produce readable behaviour oriented code.
- Cross-cutting through the different software development phases: at the planning phase, the business outcomes are mapped to behaviours, where they are then decomposed into a set of features in the analysis phase. Then at the implementation phase, the EATs take place in which testing classes are derived from scenarios.

III. Related Work

The APLE literature tackled various problems for the different RE activities (i.e., requirements elicitation, analysis, modelling, verification and validation, and management). After thoroughly studying the APLE RE literature and to the best of our knowledge, none of the previous efforts in this area proposed a RE solution that was based on BDD.

Additionally, all the attempts [10-27], except for the efforts in [28-31], focused on adopting ASD practices in already existing SPLE environments. These efforts are placed on the other spectrum of our work which is focusing on building and managing APLE in established agile-based environments.

As a further matter, there were no efforts in the literature that offered a reliable RE solution that addressed the five activities of the RE process. Although there was an all-inclusive RE solution attempt [13,14] in the literature, the authors did not validate their work through either a theoretical or a practical case study. Additionally, the authors collected their data from managers only and disregarded the perspective of the other stakeholders. Thus, directly violates the values of ASD where the perspectives of all the involved stakeholders should be taken into consideration throughout the development lifetime. Finally, none of the literature mentioned in this paper conducted a real-life empirical study to validate the respective proposed work.

The aforementioned research gaps were further confirmed by five systematic literature reviews [32-36]. These studies concluded that RE was not addressed properly or sufficiently in APLE regardless of the agility degree of the used development approach. Based on these findings and in addition to the crucial role of RE in the success of software projects, it has become imperative to have a systematic lightweight RE approach to reactively and incrementally develop and manage APLs.

IV. Summary of the BDRE Approach

The BDRE approach depends on BDD to have an incremental evolutionary flexible RE process. The full details about the BDRE approach are available in [37]. In BDRE, it is assumed that business goals, both functional and non-functional, are already identified and available for the development team to start their RE process. Generally, business goals are derived from the business need of finding solutions for a particular business problem.

The BDRE approach consists of five key activities: requirements elicitation, analysis, modelling, validation and verification, and management. Each activity is briefed as follows:

- Requirements elicitation: This is the first step in the BDRE approach where the work starts outside-in. The input to this activity is the set of solution hypotheses for the already identified business problem. The development team uses prototyping to determine the relevancy of the proposed solutions set to the underlined business goal. After agreeing on the final set of solution, the development team determines the scope of the system accordingly. After that, the development team and the customer's representative decide the initial set of features, reflecting the needed behaviour of the system-under-development (SUD), to be developed in the next iteration. This concludes the elicitation activity with that initial set of features as an output.
- Requirements analysis: This is the second activity in the BDRE approach where the initial user requirements are further examined. The initial set of features from the previous activity in addition to the already existing features, of other products in the same SPL, are fed as an input for the analysis activity. The personnel representing the roles of business analyst, developer, and quality assurance conduct specifications workshops (aka. the three Amigo's meetings) to further analyse and negotiate that given inputs. Firstly, they examine the relevancy and the clarity of the given features in comparison to the business goals. Then, they come to a consensus on which features to consider as core assets and which ones to consider as variabilities. In case they detect an abnormality in the given requirements, they may go back to the requirements analysis activity for further inspection. Otherwise, they conclude this activity by producing an initial set of user stories for each core asset/variability feature.
- Requirements modelling: This is the third step in the BDRE approach with the initial set of user stories, produced at the analysis activity, as an input. The main goal of this activity is to illustrate each user story by an example. This is achieved through developing a series of real scenarios with actual values for each user story. After meetings and negotiations, the development team finalises the initial set of scenarios (i.e., the output of this activity) for each user story of each feature. If a scenario or a user story needs further clarification, the development team may go back to the analysis activity. Otherwise, they proceed to the next step in the BDRE approach.
- Requirements validation and verification (V & V): This is the fourth step in the BDRE approach. The three Amigo's meetings take place again for refining the scenarios, produced from the modelling activity, according to their relevancy and importance. The purpose of this activity is to make sure that all the scenarios are done. To ensure that this happens, all the associated test cases of each scenario must successfully pass. Before producing the final set of scenarios, the development team negotiates and discusses all the examples with the customer's representative. In case of a disagreement, the three Amigos may decide to go back to the modelling activity or start over from the elicitation activity based on the severity level of the situation. Otherwise, the development team automates the produced final set of scenarios; thus, producing executable (aka. automated) specifications. The output of this V & V activity is the actual implementation, till the current development iteration, of the SUD.
- Requirements management: This is a cross-cutting activity in the BDRE approach through which all the other activities of the approach are maintained and managed.

V. Observational Study

This section presents an evaluation to investigate the feasibility and usefulness of the proposed BDRE approach.

A. Research Instruments

A research instrument is a tool that is used to measure, obtain, and analyse data subjects. Research instruments could be qualitative, quantitative, or a mix. In this observational study, a mixed approach seemed to be the better option as our level of understanding and familiarity with the product-under-study evolved throughout the lifetime of the development. The following are the research instruments [38] we used:

- Qualitative Methods: A qualitative research instrument is an exploratory tool that is used to have a better understanding of the subject at hand. It provides an in-depth look into the problem and/or helps developing ideas or solution hypotheses. In this research, we used two qualitative methods:
  - Observation: When using the observation research instrument, the observer can play the role of either a participant-observer or an observer participant. A participant-observer becomes a member of the community being observed; thus, enables them to earn the right to participate in the various activities accordingly. An observer participant, on the other hand, is treated as a visitor who can only observe the behaviour and the working environment of the development team, with no actual participation in their activities. Most of the time, we were an observer participant with few participations in some hands-on activities.

- Interviews: They are an integrated part of any agile-based environment. Interviews are basically a set of questions, regardless of their form (i.e., structured, semi-structured, unstructured, or a mixed-form interviews), with respective answers. Although agile advocates face-to-face communications, this might not be feasible at all times in practice. Alternatively, interviews can be mediated via telephones or other electronic means. We mainly used three types of interviews: in-depth interviews, face-to-face interviews, and discussion groups.

- Quantitative Methods: Quantitative research instruments are techniques that transform data from opinions/feelings into numbers and consequently from being subjective into being objective. One of the most popular quantitative research instruments is questionnaires. In this technique, questions can be in the format of multiple choices, dichotomous, short answers, checkboxes, drop-down, rating scales, and more. Depending on the research needs, one or more question formats can be adapted. In this research, we used the rating scale questions format. In this format, a participant is required to give an answer based on a well-defined evenly spaced range.

B. Working Environment

We tested the proposed approach in a small-sized (i.e., 100 – 200 employees) start-up agile-based company that is based in Egypt. The company has an intensive experience in agile development; in particular, Lean and Scrum agile methods.

The company focuses on the main agile practices such as iterative and incremental development; refactoring; automated testing; short iterations; pair programming; self-organising cross-functional teams; continuous deployment; progressive discovery; user story maps; and objectives and key results.

As the BDRE approach shares the same already implemented agile practices in place, the development team welcomingly embraced the proposed approach.

C. The Product under Development: RevoSuite

RevoSuite is a Business-to-Business Enterprise Software-as-a-service (SaaS). It is an artificial intelligence (AI)-enabled customer relationship management (CRM)/customer lifecycle management (CLM)/business intelligence (BI) system for pharmaceutical and life sciences businesses. The development of the product started in 2012 and evolved throughout the years. New enhancements are still added to the product despite being realised in the market late 2012.

D. The Observational Study Goal

The goal of this observational study is to investigate the feasibility of the BDRE approach in a real-life industrial case study. The elements of the observational study are inferred from the values of BDD. Table I lists the five elements of the observational study and the required observation from each one of them.

The participants in this study volunteered to take a part in our observational study. All the participants, except for the customer's representative, have worked on RevoSuite throughout its lifetime. The total number of volunteering participants is 24, categorised as follows: six business analysts, eleven developers, six quality assurance, and one customer's representative.

Prior to starting the observational study, we explained the BDRE approach to the participants and offered them training on how to implement the approach. Afterwards, the participants took parts in various complexity pilot projects throughout the RevoSuite different development iterations. Thus, enabled us to monitor and observe the participants' performance. Additionally, we developed a questionnaire addressing the elements listed in Table I in further details and asked our participants to anonymously answer the questionnaire from the perspective of each one's role.

TABLE I. OBSERVATIONAL STUDY ELEMENTS

Study Element	Required Observation
Learnability	Whether the participants are able to use the BDD ubiquitous language to express features, user stories, and scenarios
Coherence	Whether the participants are able to produce consistent outputs compared to that of the required business goals
Restrictions/Conflicts	Whether the participants are able to find all the explicit and implicit constraints and conflicts through executable specifications
Evolution	Whether the participants are able to start a feature, integrate new changes as they come in, and eventually deliver the feature in a manner consistent with the behaviour expected by the customer.
Readability	Whether the participants are able to read and understand the documentation, including the code, of the system.

VI. RESULTS AND DISCUSSION

This section presents and discusses the results of both the pilot projects and the questionnaire.

A. Pilot Projects Results

The participants' performance was measured by two factors: the time spent on each feature from beginning to end; and the uniformity of their output compared to that expected by the respective business goal. In general, the time spent on each feature was directly proportional to the complexity degree of that feature. Consequently, the time spent in high-complexity pilot projects varied between double to tribble that of the low-complexity projects. Despite that, the performance of all the participants was almost consistent regardless of the complexity of the features. The only exception was for the one customer's representative whose performance was inversely proportional to the complexity of the feature at hand.

In projects with low-medium complexity, we observed that:

- Learnability: almost all the participants were able to successfully use the BDD ubiquitous language to illustrate features, user stories, and scenarios.
- Coherence: more than 80% of the participants were able to have consistent outputs to those of the required business goals.
- Restrictions and conflicts: more than 75% of the participants were able to deduce all the explicit restrictions and conflicts. However, only half of them were able to spot all the implicit constraints.
- Evolution: more than 80% of the participants were able to start a feature, integrate new changes as they merge, and eventually deliver the feature (i.e., a core asset or a variability) in consistency with the expected behaviour of the system.
- Readability: all the participants were able to read and understand the system's documentation with minor difficulties.

In projects with high complexity, on the other hand, the participants spent more time on the features although they attained the same performance as that of the low-medium complexity projects. The only exception was the customer's representative whose performance dropped as the complexity of the feature increased.

B. Questionnaire Results

We used a five points Likert-scale, ranging from strongly disagree to strongly agree, to record the questionnaire responses. Fig. 1 illustrates the average responses per role for each question in the questionnaire. According to the recorded responses, the participants have come to a consensus that the BDRE approach is flexible, easy to understand, and easy to apply in practice. Some participants, however, shared their concerns about the potentiality and reliability of the BDRE approach in terms of scalability or when used with more complex systems. Lastly, finding implicit constraints was tricky and out of the comfort zone for some developers as well as for the customer's representative.

VII. Conclusion

APLE is increasingly gaining momentum in software development. Nonetheless, adopting APLE in practice calls for a special focus on RE. We proposed the BDRE approach to provide a flexible lightweight incremental RE process through using BDD throughout the different activities of RE. In this paper, we presented an observational study to examine five aspects of the BDRE approach in an empirical real case study. The results of the study were encouraging and shed the light on the strengths and weaknesses of the approach.

Acknowledgment

The authors would like to thank RevoSuite Company for their guidance and cooperation throughout the conduction of the observational study presented in this paper.

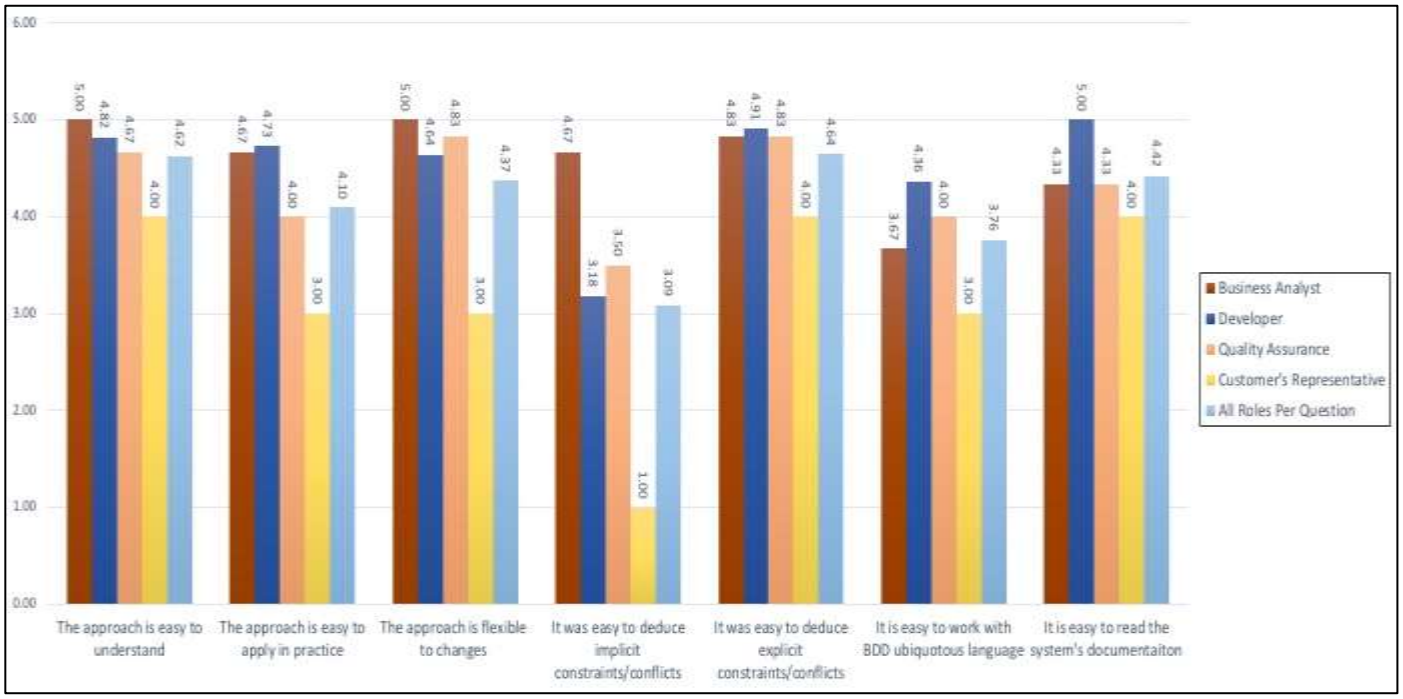

Fig. 1. Average Responses per Role to the Likert-Scale Questions.

REFERENCES

[1] K. Cooper, X. Franch, "APLE First International Workshop on Agile Product Line Engineering". IEEE Computer Society, pp. 205–206. Silver Spring, USA, 2006.

[2] K. Pohl, G. Böckle, F. Linden, Software Product Line Engineering: Foundations, Principles and Techniques. Springer, Germany 2005.

[3] L. Williams, A. Cockburn, Agile Software Development: It's About Feedback and Change, Computer 36(6), 39–43, 2003.

[4] D. North, Introducing BDD Available at: http://dannorth.net/introducing-bdd, 2006, last accessed 2018/1/1

[5] L. Westfall, Software Requirements Engineering: What, Why, Who, When and How?, Retrieved from http://www.westfallteam.com/Papers/The_Why_What_Who_When_and_How_Of_Software_Requirements.pdf, last accessed 2018/11/12

[6] Standish Group, The CHAOS Report 2015, Retrieved from https://www.standishgroup.com/sample_research, last accessed 2015/10/2

[7] G. Kontonya, I Sommerville, Requirements Engineering: Processes and Techniques. John Wiley & Sons, 1998.

[8] M. Chemuturi, Requirements Engineering and Management for Software Development Projects. Springer Publishing Company, 2012.

[9] D. Astels, A new look at test driven development. http://techblog.daveastels.com/files/BDD_Intro.pdf, last accessed 2019/3/1

[10] M.A. Noor, R. Rabiser, P. Grünbacher, "A collaborative approach for reengineering-based product line scoping", In: the 1st International Workshop on Agile Product Line Engineering (In conjunction with SPLC), 2006.

[11] M.A. Noor, R. Rabiser, P. Grünbacher, Agile Product Line Planning: A Collaborative Approach and a Case Study, Journal of Systems and Software, Vol. 81(6), pp. 868–882, 2007.

[12] M.A. Noor, P. Grünbacher, C. Hoyer, "A collaborative method for reuse potential assessment in reengineering-based product line adoption", In: Balancing Agility and Formalism in Software Engineering. LNCS, vol. 5082, pp. 69–83. Springer, Heidelberg, 2008.

[13] K. Feng, M. Lempert, Y. Tang, K. Tian, K. Cooper, X. Franch, "Developing a survey to collect expertise in agile product line requirements engineering", In: Agile 2007 Conference, International Research-in-Progress Workshop on Agile Software Engineering, pp. 1–4, 2007.

[14] K. Feng, Towards an Agile Product Line Requirements Engineering Framework: Knowledge Acquisition and Process Definition, Ph.D. Dissertation, The University of Texas at Dallas, 2009.

[15] P. Trinidad, D. Benavides, A. Durán, A. Ruiz-Cortés, M. Toro, Automated Error Analysis for the Agilization of Feature Modeling, Journal of Systems and Software, vol. 81(6), pp. 883–896, 2008 .

[16] R. Paige, X. Wang, Z. Stephenson, P. Brooke, "Towards an agile process for building software product lines", In: Proceedings of the 7th International Extreme Programming and Agile Processes in Software Engineering, Springer, Heidelberg, pp.198–199, 2006.

[17] G. Kakarontzas, I. Stamelos, P. Katsaros, "Product line variability with elastic components and test-driven development", In: Proceedings of the 2008 International Conference on Computational Intelligence for Modelling Control and Automation, IEEE Computer Society. 146–151, 2006.

[18] M. Raatikainen, K. Rautiainen, V. Myllärniemi, T. Männistö, "Integrating product family modeling with development management in agile methods", In:Proceedings of the 1st International Workshop on Software Development Governance, pp. 17–20, 2008.

[19] L. Taborda, The Release Matrix for Component-Based Software Systems, In: Proceedings of Component-Based Software Engineering, LNCS, vol. 3054, pp. 100–113, Springer, Heidelberg, 2004.

[20] R. Kurmann, "Agile SPL-SCM agile software product line configuration and release management", In: 1st International Workshop on Agile Product Line Engineering (In conjunction with SPLC), 2006.

[21] R. Carbon, M. Lindvall, D. Muthig, P. Costa, "Integrating product line engineering and agile methods: flexible design up-front VS. incremental design", In: Proceedings of the 1st International Workshop on Agile Product Line Engineering In conjunction with SPLC, pp. 1–8, 2006.

[22] R. Carbon, J. Knodel, D. Muthig, G. Meier, "Providing feedback from application to family engineering – The product line planning game at the Testo AG", In: Proceedings of the 12th International Software Product Line Conference, IEEE Computer Society, pp. 180–189, 2008.

[23] P. O'Leary, M.A. Babar, S. Thiel, I. Richardson, "Product derivation process and agile approaches: exploring the integration potential", In: Proceedings of the 2nd IFIP Central and East European Conference on Software Engineering Techniques. pp. 166–171, 2007.

[24] P. O'Leary, M.A. Babar, S. Thiel, I. Richardson, "Towards agile product derivation in software product line engineering", In: Proceedings of the 4th International Workshop on Rapid Integration of Software Engineering Techniques, pp. 19–32, 2007.

[25] P. O'Leary, S. Thiel, G. Botterweck, I. Richardson, "Towards a product derivation process framework", In: Proceedings of the 3rd IFIP TC2 Central and East European Conference on Software Engineering Techniques, pp. 189–202, 2008.

[26] P. O'Leary, F. McCaffery, I. Richardson, S. Thiel, "Towards agile product derivation in software product line engineering", In: Proceedings of the 16th European Conference on Software Process Improvement, pp. 81–86, 2009.

[27] P. O'Leary, R. Rabiser, I. Richardson, S. Thiel, "Important issues and aey Activities in product derivation: experiences from two independent research projects", In: Proceedings of the 13th International Software Product Line Conference, pp. 121–130, 2009.

[28] Y. Ghanam, S. Park, F. Maurer, "A test-driven approach to establishing and managing agile product lines", In: Proceedings of the 5th Software Product Lines Testing Workshop in conjunction with SPLC'08, pp. 151–156, 2008.

[29] Y. Ghanam, F. Maurer, "An iterative model for agile product line engineering", In: The SPLC Doctoral Symposium, 2008 - in conjunction with the SPLC'08, pp. 377–384, 2008.

[30] Y. Ghanam, F. Maurer, "Extreme product line engineering: managing variability and traceability via executable specifications", In: Agile Conference, AGILE '09, IEEE Computer Society, pp. 41–4, 2009.

[31] Y. Ghanam, F. Maurer, "Extreme product line engineering refactoring for variability: a test-driven approach", In: Proceedings of 11th International IV Conference on Agile Processes in Software Engineering and Extreme Programming, XP 2010, LNBIP, vol. 48, pp. 43–57, Springer, Heidelberg, 2010.

[32] F. F. Farahani, R. Ramsin, "Methodologies for agile product line engineering: a survey and evaluation", In: Proceedings of the 13th International Conference SoMeT_14, Amsterdam: IOS Press BV, pp.545-564, 2014.

[33] I.F. da Silva, P. Neto, P. O'Leary, E. de Almeida, S.R. de Lemos Meira, Agile Software Product Lines: A Systematic Mapping Study, Software: Practice and Experience, vol. 41(8), pp. 899–920, 2011.

[34] J. Díaz, J. Pérez, P.P. Alarcón, J. Garbajosa, Agile Product Line Engineering – A Systematic Literature Review, In: Software Practice and Experience, vol. 41(8), pp. 921–941, 2011.

[35] V. Alves, N. Niu, C. Alves, G. Valença, Requirements Engineering for Software Product Lines: A Systematic Literature Review, In: Information and Software Technology, vol. 52(8), pp. 806–820, 2010.

[36] D.F.S. Neiva, RiPLE-RE: A Requirements Engineering Process for Software Product Lines, M.Sc. Dissertation, Universidade Federal de Pernambuco, Brazil, 2009.

[37] H. Elshandidy, "Behaviour-driven requirements engineering for agile product line engineering", In: Proceedings of the 2019 IEEE 27th International Requirements Engineering Conference (RE), Jeju Island, Korea (South), pp. 434-439, 2019.

[38] R. Trigueros, M. Juan, F. Sandoval, Qualitative and Quantitative Research Instruments Research tools, 2017.