SOFTWARE – PRACTICE AND EXPERIENCE
Softw. Pract. Exper. 2011; **41**:899–920
Published online 3 May 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/spe.1078

Agile software product lines: a systematic mapping study

Ivonei Freitas da Silva[1,3,*,†], Paulo Anselmo da Mota Silveira Neto[1,3], Pádraig O'Leary[2,3], Eduardo Santana de Almeida[2,3] and Silvio Romero de Lemos Meira[1]

[1]*Informatics Center, Federal University of Pernambuco, Recife, PE, Brazil*
[2]*Computer Science Department, Federal University of Bahia, Salvador, BA, Brazil*
[3]*RiSE (Reuse in Software Engineering), Recife, PE, Brazil*

SUMMARY

Background: Software product lines and Agile methods have been an effective solution for dealing with the growing complexity of software and handling competitive needs of software organizations. They also share common goals, such as improving productivity, reducing time-to-market, decreasing development costs and increasing customer satisfaction. There has been growing interest in whether the integration of Agile and SPL could provide further benefits and solve many of the outstanding issues surrounding software development. *Objective*: This study investigates the state-of-the-art in Agile SPL approaches, while identifying gaps in current research and synthesizing available evidence. It also provides a basis for a deeper understanding of the issues involved in the integration of Agile and SPL. *Method*: A mapping study was undertaken to analyze the relation between Agile and SPL methods. A set of four research questions were defined in which the 32 primary studies were evaluated. *Results*: This study provides insights into the integration of Agile and SPL approaches, it identifies the current gaps in the research, synthesize the available evidence and propose specific Agile methods and practices for integration in SPL. *Conclusions*: In general, few studies describe the underlying Agile principles being adopted by proposed Agile SPL solutions. The most common Agile practices proposed by the studies came from the XP and Scrum methods, particularly in the pro-active SPL strategy. We identify certain Agile methods that are being overlooked by the Agile SPL community, and propose specific SPL practices areas suitable for adoption of Agile practices. Copyright © 2011 John Wiley & Sons, Ltd.

Received 11 August 2010; Revised 29 November 2010; Accepted 22 February 2011

KEY WORDS: systematic mapping study; Agile methods; software product lines; Agile principles

1. INTRODUCTION

Software development is a complex composition of processes, technologies, tools and people, often in domains with volatile requirements. Furthermore, with the growing demand for higher quality software, reduced development costs and shorter time-to-market, the idea of leveraging the strengths of existing approaches has been raised as a possible solution. Whereas *Software Product Lines* (*SPL*) engineering provides a systematic approach to reuse common artifacts to facilitate the delivery of different products [1]; *Agile methods* focus on working code while reducing up-front design and process overhead of traditional approaches [2], through the adoption of fundamental values and principles [3]. Despite the differences in how both approaches perform software development, they propose to deliver common benefits, such as improving productivity, reducing time-to-market,

*Correspondence to: Ivonei Freitas da Silva, Informatics Center, Federal University of Pernambuco, Recife, PE, Brazil.
†E-mail: ifsse3@gmail.com

decreasing development costs, and increasing customer satisfaction. By leveraging these common benefits, a balance of Agile and SPL could provide interesting possibilities for a new approach to handle customer changes, manage variability in products, and leverage software reuse opportunities. However, there are several challenges for the integration of Agile in SPL due to differences in the philosophies of both approaches, such as design and change management strategies [4, 5]. Moreover, Agile methods do not propose to develop flexible artifacts for reuse [4, 6] or develop and maintain extensive documentation as required by SPL [7]. Furthermore, in general, the SPL approach uses two separate distinct processes, a domain and application engineering process [7], and focuses on architecture and business, with well-defined activities and roles. On the other hand, Agile methods advocate a light process, with no systematic reuse, and a focus on people, with few defined activities and roles [2].

Considering these possibilities and challenges of Agile SPL, and due to the growing body of knowledge on Agile SPL integration, a systematic mapping study on the integration of Agile and SPL is timely. This Mapping Study (MS) based on [8, 9], aims to collect evidence about how Agile SPL research is structured, synthesize current evidence on the integration of both approaches and identify further challenges for the integration of Agile methods and SPL.

The remainder of this paper is organized as follows: Section 2 describes the systematic mapping process undertaken in this study. Section 3 presents the main findings of the study and analyzes the findings. Section 4 considers the threats to the validity of this research. Finally, Section 5 presents a summary of the work and the directions for future work.

2. SYSTEMATIC MAPPING STUDY PROCESS

While Systematic Reviews (SR) are commonly [8, 10] applied to identify, evaluate, and compare all available research related to a particular research question, an MS intends to map out the research undertaken rather than to answer detailed research question [11]. An MS was used in this research to ‘map out’ the Agile SPL area and structures published the research reports and results by categorizing them [9].

In this study, we adapted some SR concepts such as the protocol definition to drive the research during the study definition improving its reliability, by providing information and adjustments that need to be validated before the MS execution. Figure 1 summarizes the steps performed to conduct this MS.

Before starting the MS protocol, an overview of Agile methods and SPL was performed on the literature in order to understand the terminology, concepts, issues, and challenges. In addition to the overview, meetings and brain-storming sessions were performed with different researchers in the area. All these data collection and discussions served as the basis for the protocol definition and the development of research questions.

2.1. Research questions

The general question that guides this mapping study and reflects our aim is: *What is the available evidence regarding the integration of agile methods*, *principles, and practices in software product lines*? To answer this question, a set of four sub-questions was derived from the primary question.

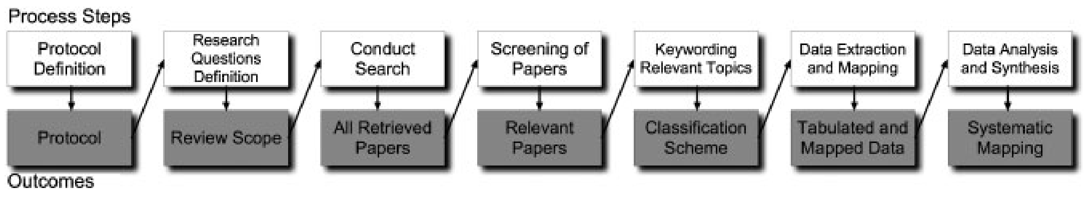

Figure 1. Systematic mapping study process, adapted from [9].

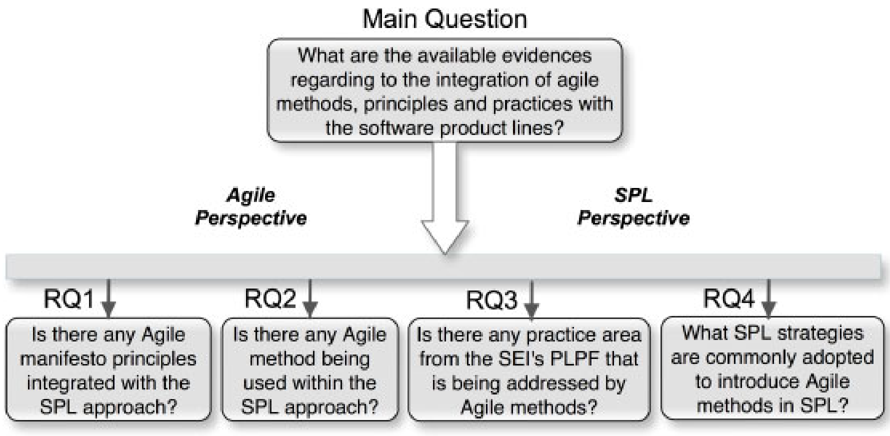
Main Question
What are the available evidences regarding to the integration of agile methods, principles and practices with the software product lines?
Agile Perspective
SPL Perspective
RQ1
Is there any Agile manifesto principles integrated with the SPL approach?
RQ2
Is there any Agile method being used within the SPL approach?
RQ3
Is there any practice area from the SEI's PLPF that is being addressed by Agile methods?
RQ4
What SPL strategies are commonly adopted to introduce Agile methods in SPL?

Figure 2. Mapping study questions.

The aim of these research sub-questions is to collect information about the adoption of agile methods in SPL. Figure 2 summarizes the questions.

Q1. *Are there any Agile manifesto principles integrated with the SPL approach*? The goal is to investigate the commonalities between Agile and SPL. If Agile manifesto principles are currently being applied within SPL, then theoretically there is potential for the integration of Agile methods and SPL.

Q2. *Is there any Agile method being used within the SPL approach*? This question investigates evidence related to the adoption of Agile methods in the SPL context. The adoption of Agile methods, fully or partially, will indicate whether and to what degree the challenges of Agile integration in SPL are being overcome.

Q3. *Are there any practice areas from the PLPF that are being addressed by Agile methods*? The SEI Product Line Practice Framework (PLPF) [1] is considered an industry standard on SPL practice. It identifies the fundamental concepts underlying SPL and the activities to be considered when creating a product line, through 29 practice areas. Each of these areas is composed of a set of tasks that an organization should perform to successfully carry out the essential work of a product line [1]. Moreover, the framework has had widespread use in organizations with different structures and business domains. Thus, this question aims to investigate each one of the PLPF practice areas, and identify which are being addressed by Agile methods.

Q4. *What SPL strategies are commonly adopted to introduce Agile methods in SPL*? According to McGregor [12], there are three main strategies to be applied in SPL development. The pro-active approach, where the core assets are developed prior to any product development; the reactive approach, where the core assets are harvested from products after they are built and deployed; and the incremental approach, where a compromise between the other strategies is reached. Neither of the three strategies is the best in all cases, the optimal strategy depends on the context in which the product line organization operates. Thus, this question will investigate whether Agile methods are applied more often in a particular SPL strategy, and investigate if and for what reasons a strategy is more suited to Agile integration.

2.2. *Conduct the search*

The search terms adopted in this study were defined using the approach described in [13], which describes five steps: (i) derive terms from the questions; (ii) identify alternative spellings and synonyms; (iii) keyword checking; (iv) use OR; and (v) AND operators to merge the search strings. The complete list of search strings is available in Table I.

Table I. Search strings.

agile **OR** 'extreme programming' **OR** scrum **OR** 'dynamic systems development method' **OR** 'crystal clear' **OR** 'feature-driven development' **OR** lean **OR** 'adaptive software development' **OR** evo **OR** 'test-driven development' **OR** 'iterative and incremental development' **OR** 'evolutionary and adaptive development'
AND
'software product line' **OR** 'software product family' **OR** 'product line engineering' **OR** 'software reuse' **OR** 'domain engineering' **OR** 'application engineering' **OR** 'core asset'

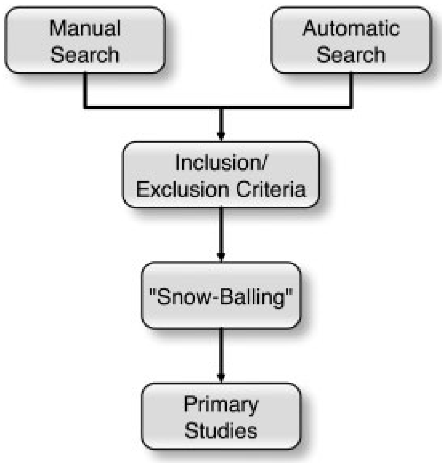

Figure 3. Search process workflow.

The search process was performed using three steps: (i) the search strings in Table I were calibrated and applied in each digital database (see Appendix A); (ii) a manual search was performed in the main Journals and Conferences (see Appendix B) proceedings in the area, covering the years 2009 and 2010. The previous years are indexed by the digital databases used in this study; (iii) and finally, the 'snow-balling' process [11], in which the references of the identified papers are analyzed, was used. Figure 3 shows the search process as a whole.

2.3. *Selection criteria*

In order to filter the studies and eliminate those that were irrelevant, i.e. do not address the research questions, we adopted the following inclusion and exclusion criteria.

Inclusion: The study must explore the practice, theory, approaches, or issues related to the integration of Agile methods in SPL. In addition, the study must be unique, i.e. when a study has been published in more than one venue, the most complete version will be used. It is important to highlight that studies, which focus on CMMI, RUP, Model-Driven Development and Open Source with contributions for an Agile SPL scenario are considered in this study.

Exclusion: We exclude those studies that do not address the integration of Agile methods and SPL. Moreover, studies that mention Agile integration in SPL, but do not discuss this integration are also excluded. Finally, studies that are only available as abstracts or PowerPoint presentations are excluded.

2.4. *Screening of papers*

The selection of studies involved a screening process composed of three filters. The aim of this screening process was to select the most suitable set of primary studies. Figure 4 shows the considerations of each filter.

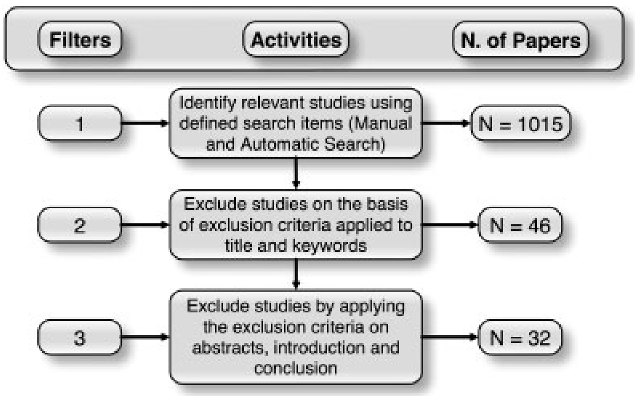
Filters
Activities
N. of Papers
1
Identify relevant studies using defined search items (Manual and Automatic Search)
N = 1015
2
Exclude studies on the basis of exclusion criteria applied to title and keywords
N = 46
3
Exclude studies by applying the exclusion criteria on abstracts, introduction and conclusion
N = 32

Figure 4. Filters of the selection process.

After performing the search string on the selected digital databases and a manual search on the main conferences and journals in the area, a total of 1015 studies were identified. The inclusion and exclusion criteria were applied on the title and keywords of the identified studies, resulting in 46 studies being selected. The large number of duplicated studies and the automatic search within the paper context of the digital database resulted in a large discrepancy between the total number of studies (1015) and resulting studies (46). Finally, a more detailed review of the abstract and in some situations, of the introduction and conclusion of the papers was performed. After this review, 32 studies remained. Two researchers authors of this paper validated the final list of studies.

2.5. *Classification scheme*

The idea of categorizing the studies was based on [9]. This classification scheme helped to categorize the results of this study. Our classification scheme involved revisiting the abstract, titles, and keywords of the selected primary studies to identify different facets within the selection. The defined facets were:

- *Research type*: Validation Research, Evaluation Research, Solution Proposal, Philosophical Papers, Opinion Papers, and Experience Papers [14] (a description of each category is presented in Table II).
- *Agile methods*: Extreme Programming, SCRUM, Feature-Driven Development, Crystal, Dynamic Systems Development Method, Lean, Adaptive Software Development, EVO, Test-Driven Development, and mixing of them.
- *Agile Manifesto principles* [3]:
    1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
    2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
    3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
    4. Business people and developers must work together daily throughout the project.
    5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
    6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
    7. Working software is the primary measure of progress.
    8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
    9. Continuous attention to technical excellence and good design enhances agility.

Table II. Research type facet [14].

Classes	Description
Validation Research	Techniques investigated are novel and have not yet been implemented in practice. Techniques used are, for example, experiments i.e. work done in the lab
Evaluation Research	Techniques are implemented in practice and an evaluation of the technique is conducted. Implementation of the technique is shown in practice (solution implementation) and the consequences of the implementation in terms of benefits and drawbacks (implementation evaluation) are demonstrated
Solution Proposal	A solution for a problem is proposed, the solution can be either novel or a significant extension of an existing technique. The potential benefits and the applicability of the solution is shown by a small example or a good line of argumentation
Philosophical Papers	These papers sketch a new way of looking at existing things by structuring the field in form of a taxonomy or conceptual framework
Opinion Papers	These papers express the personal opinion of somebody whether a certain technique is good or bad, or how things should been done. They do not rely on related work and research methodologies
Experience Papers	Experience papers explain what and how something has been done in practice. It has to be the personal experience of the author

10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

- *Contribution type*: Process, Method, Model, Framework, and Metric.
- *SPL strategies*: Pro-active, Reactive, and Incremental [12].
- *PLPF* [1][‡]: Architecture Definition, Architecture Evaluation, Component Development, Mining Existing Assets, Requirements Engineering, Software System Integration, Testing, Understanding Relevant Domains, Using Externally Available Software, Configuration Management, Make/Buy/Mine/Commission Analysis, Measurement and Tracking, Process Discipline, Scoping, Technical Planning, Technical Risk Management, and Tool Support.

2.6. Data extraction

A data extraction form was designed in order to gather the required information to address the objectives of this study. To collect the required data, the full paper was read and the following information were extracted from each study: *Objectives*; *Problems*; *Results*; *Answers for each research question*; *and information needed to categorize the study using the previously defined facets*.

During the extraction, some studies could be classified in more than one category. Thus, primary studies can appear more than once. For example, in [15], two PLPF areas are considered: *requirements engineering* and *testing*.

3. OUTCOMES

In this section, we describe the main findings of the data extraction activities, in which the results were tabulated based on the previously defined facets (see Section 2.5). To further visualize the results, we grouped them according to each research question. The tables and figures

[‡]As Agile methods do not consider the Organizational Management practice area, it is not considered as a facet.

contain the number of studies that can be classified according to a particular classification category.

3.1. Results

In the following subsections, we present the main findings of each research topic, the relationship between Agile and SPL approaches, and the classification scheme. It is important to mention that as it is a recent research area (Agile SPL), the findings should not be considered as the absolute truth, but initial evidences or tendencies.

Some domains for the integration of Agile and SPL approaches were identified, such as air traffic control, medical, project management, e-commerce, e-mail management, automotive, microwave, and embedded systems. However, the case studies and experiments reported did not present the domain size and complexity and protocols to guide the validation or evaluation of the study proposal.

3.1.1. RQ1—agile manifesto principles and SPL. When performing the analysis, few studies explicitly stated the Agile principle that was being applied in the study. In most cases, the principle being applied was deduced from the Agile practices identified. Using this deduction, we noticed that all Agile principles were addressed by at least four studies (see Table III). It is possible that the SPL approaches have already addressed those principles, but with different practices, perspectives, and contexts [6, 12, 24, 36]. No evidence was found on the application of one Agile principle forcing the use of another.

In 25% of the studies [5, 37–43], the focus was on agile methods without clearly mentioning its practices, making it difficult to identify which agile principle(s) is/are being addressed by the study.

In [6, 12, 20] no empirical evidence is presented and these studies were classified as opinion papers and addressed most of the Agile principles within the PLPF practice areas.

All the Agile principles were addressed by studies that presented empirical evidence, especially, *P1*, *P9*, and *P10* (see Table III). *P1*, *P9*, and *P10* were exploited through Agile practices, such as, test-driven development (TDD) in [15, 18, 19, 30–32, 34]. *P4* principle has been applied on studies (Table III) that exploit the initial phases of SPL, specifically, scoping and requirements. The studies reinforce that it is important to have interaction between business stakeholders including customers and development stakeholders. All of these interactions help to provide rapid feedback, better business alignment, establish common knowledge and goals, and focus on the value-adding features.

Analyzing the relationship between Agile principles and SPL, we can make some deductions: the Agile principle *P1* is partially satisfied by SPL as both aim to achieve shorter time-to-market. However, despite this shared principle, they use different strategies to achieve this goal. Whereas SPL addresses it by introducing continuous iterations during application engineering phase [24], Agile approaches use continuous integration, short interactions, incremental deployment [16, 17, 21, 22, 25–27] and mainly test-driven development [18–20, 23, 24].

Automated support is a necessity for managing the complexity and variability inherent in software product lines and automated development approaches facilitate the Agile Principle *P2*. Automated techniques allow product teams to implement changing customer requirements late in the development life cycle and automation enables these changes to be implemented quickly [24].

Regarding *P10* principle, it is addressed by different Agile practices such as Test-Driven Development (TDD) [30], planning game [4], incremental design [4] and techniques from collaboration engineering [27, 29]. These Agile practices are applied in the Testing and Scoping PLPF areas. Both Agile and SPL practices advocate focusing on what is essential for implementation. The integration of principle *P5* was empirically demonstrated through collaboration engineering in the SPL planning, FDD, and DSDM agile methods [25–27]. Finally, *P8* principle is exploited through the FDD and DSDM agile methods [25, 26].

Analyzing the studies that have addressed the Agile principles and discussed how they have performed it, we have crossed these facets (see Figure 5) to identify which principles have been evaluated in the industry, and which principles do not have any empirical evidence.

Table III. Agile manifesto principles.

ID	Principles	Studies	Number of studies
P1	'… early and continuous delivery …'	[6, 12, 16–26]	13
P2	'Welcome changing requirements…'	[6, 12, 17, 19, 21, 24–27]	9
P3	'… shorter timescale …'	[6, 12, 16, 19–21, 25, 26]	8
P4	'Business people and developers together …'	[4, 6, 12, 17, 20, 21, 25, 26, 28, 29]	10
P5	'Build projects around motivated individuals …'	[6, 12, 25–27]	5
P6	'… face-to-face conversation'	[6, 12, 17, 20, 25–28]	8
P7	'Working software is the measure of progress'	[4, 6, 12, 19, 21, 25, 26, 30]	8
P8	'… sustainable development … constant pace …'	[6, 12, 25, 26]	4
P9	'… technical excellence and good design …'	[6, 12, 15, 18–20, 23, 25–27, 30–34]	15
P10	'Simplicity …'	[4, 6, 12, 15, 18–20, 25–27, 30–32, 34, 35]	15
P11	'The best architecture, requirements, and designs emerge from self-organizing teams'	[4, 6, 12, 18, 19, 25, 26, 30]	8
P12	'… team tunes and adjusts its behavior accordingly'	[6, 12, 18–21, 25–27]	9

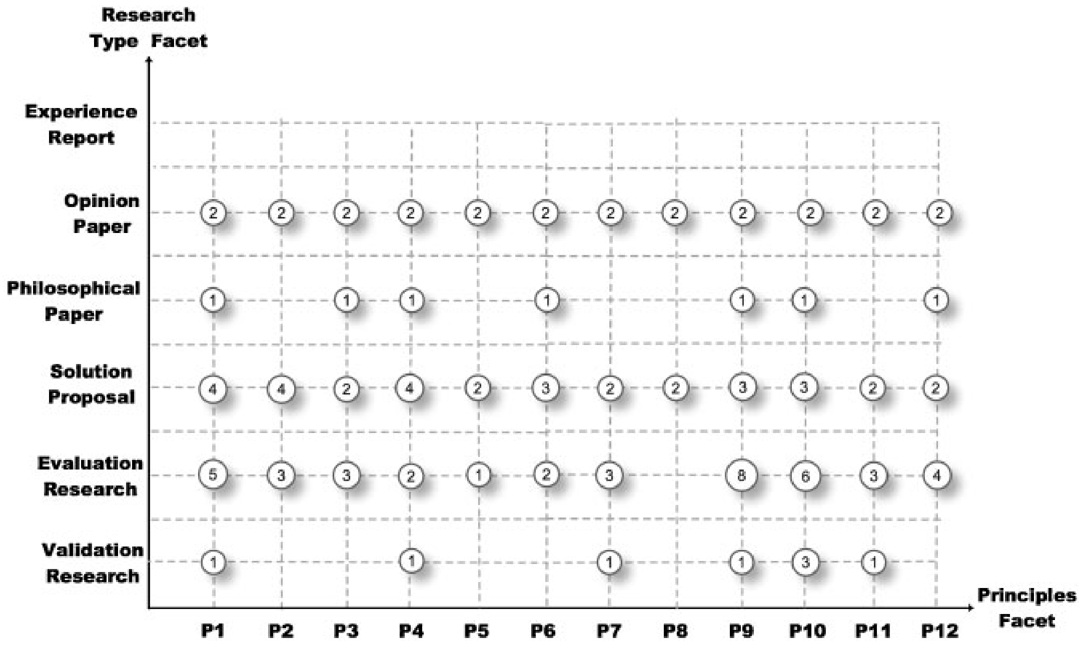

Figure 5. Agile principles versus research types.

Table IV. Number of studies versus agile methods.

Methods	References	Number of studies
Extreme Programming	[4, 16, 17, 19, 20, 24, 26, 28, 31, 40]	10
SCRUM	[16–18, 20, 23, 41]	6
Feature-Driven Development	[25, 43]	2
Crystal	None	0
Dynamic Systems Development Method	[5, 26]	2
Lean	[17]	1
Adaptive System Development	None	0
Evo	[21, 33]	2
Test-Driven Development	[8, 15, 18–20, 23, 24, 30, 31, 34, 35]	11
Combination	[16–20, 23, 24, 26, 31]	9

3.1.2. RQ2—agile methods used within SPL. Agile methods apply time boxed iterative and evolutionary development, adaptive planning, promote evolutionary delivery, and include other values and practices that encourage feedback and flexible response for changes. Some practices are shared among Agile methods, such as *short time boxed iterations* and *adaptive plans*.

In addition, Agile methods promote practices, principles, and values that reflect the Agile Manifesto [3, 44]. Table IV shows no evidence of the application of some Agile methods in SPL, such as Crystal and Adaptive Systems Development (ASD).

This indicates that some Agile practices and methods for SPL are being ignored. For example, the DSDM method focuses on business domains similar to the Scoping practice area in SPL. In addition, Lean is only addressed by [17], through adoption of the Agile practice *white board* as part of the framework for agile product line requirements engineering process [17]. This study also describes others Agile methods, such as Extreme Programming, Scrum, and Agile Model-Driven Development (AMDD) [45]—an agile version of Model-Driven Development (MDD) [46].

The study [33] reports an industrial case study, by applying Evo method, where it is defined a process whose needs are strategic (long-term objectives of the organization, SPL), tactical (medium-term objectives as specific product, Agile), and operational (short-term, day-to-day operations).

On a strategic level, it uses bi-annual releases with product development and general market-oriented activities (sales, marketing, and so on). On a tactical level, the case study uses the Evo method for software development. On an operational level, the case study performs day-to-day activities with the customers use of the product in mind.

In [21], the EVO method is adopted by an SPL organization using a two-step approach. First, the system needs must be restructured to establish a level of maintainability that is manageable. Second, an Agile process must be complemented with continuous semi-automated quality monitoring and refactoring support. They summarize that Evo is an example of how to build 'your home ground' [47] for Agile SPL.

The Feature-Driven Development (FDD) method appears to be the most suitable Agile method for SPL, due to its use of features and claim of reuse practices. However, few studies consider its application within SPL. An exception is [25], which proposes mixing FDD with SPL. The proposed approach though derived from FDD, could be considered more Agile, if we consider the potential higher reuse levels and flexibility gains from the SPL approach.

In [43], the authors combine SPL and Agile Methods by decomposing customer requirements in terms of features. Thus, they use FDD to represent these features and provide an automatic way to perform feature model error analysis, thus improving agility.

DSDM is another method which advocates reuse practices. In [5], the authors report that the definition and maintenance of software product line architecture could be made more Agile by adoption of DSDM. This can be achieved as DSDM is plan-driven and puts more emphasis on the development of a software architecture.

The study [26] reports a solution proposal by mixing DSDM and XP within SPL and highlights conflicts when integrating Agile practices in different areas, such as requirements, code ownership (XP practice), customer relationships, planning for dependencies, and documentation.

SCRUM, XP, and TDD are the most frequently used in the SPL context. In [16–18, 20, 23, 41], some SCRUM practices are applied, such as *backlog item*, *sprint*, *stand up daily meeting*, *retrospective*, and *reflection*. These practices are used in many contexts, for example, in [16] the authors exploit SCRUM to assemble the SPL reference architecture as an exploratory phase between domain engineering and application engineering. In addition, SCRUM and other Agile methods can also be used during the application engineering phase to build products. [18] exploits SCRUM in a reactive bottom-up SPL strategy where reusable artifacts are extracted from existing assets on demand.

Some studies [16, 20, 23] show that XP and SCRUM can be combined for use in the SPL context. The popularity of this combination can be due to the contrasting focus. SCRUM is focused on project management, while XP is focused on development. Thus, together they form a complementary combination.

According to Read [48], TDD supports defect reduction and it can act as a source of documentation and a form of requirements specification. It is commonly applied on the variability management and also to link feature models with code using a reactive strategy. In [18, 15, 19, 30–32], the studies performs refactoring in artifacts and code to get variability.

Different types of contributions are made by different Agile methods in SPL. While none of the studies defined explicitly the contribution type, a classification was made based on the study descriptions. XP [24, 26, 28, 38], FDD [25], DSDM [26], Evo [33], and TDD [24, 30] methods make a process contribution. In [30, 34, 35], a method contribution is presented, whereas model contribution type is made by XP [31], Scrum [18], and TDD [18, 31]. Finally, a framework contribution is made by XP [4, 17, 19, 20], Scrum [17, 20], FDD [43], Lean [17], and TDD [15, 19, 20] (as shown in Figure 6). No Agile SPL study makes a contribution to metrics.

In Figure 6, 11 studies [16, 18, 19, 21, 23, 28, 30–33, 43], including 6 combinations (XP and SCRUM [16], TDD and SCRUM [18, 23] and XP and TDD [19, 31, 32]), have been assessed through evaluation, one study described an experience report [38], and another three described a validation [4, 15, 35] using academia or industry environment.

When merging Agile methods with the solution proposal category, six studies [17, 24–26, 34, 41] used this research type.

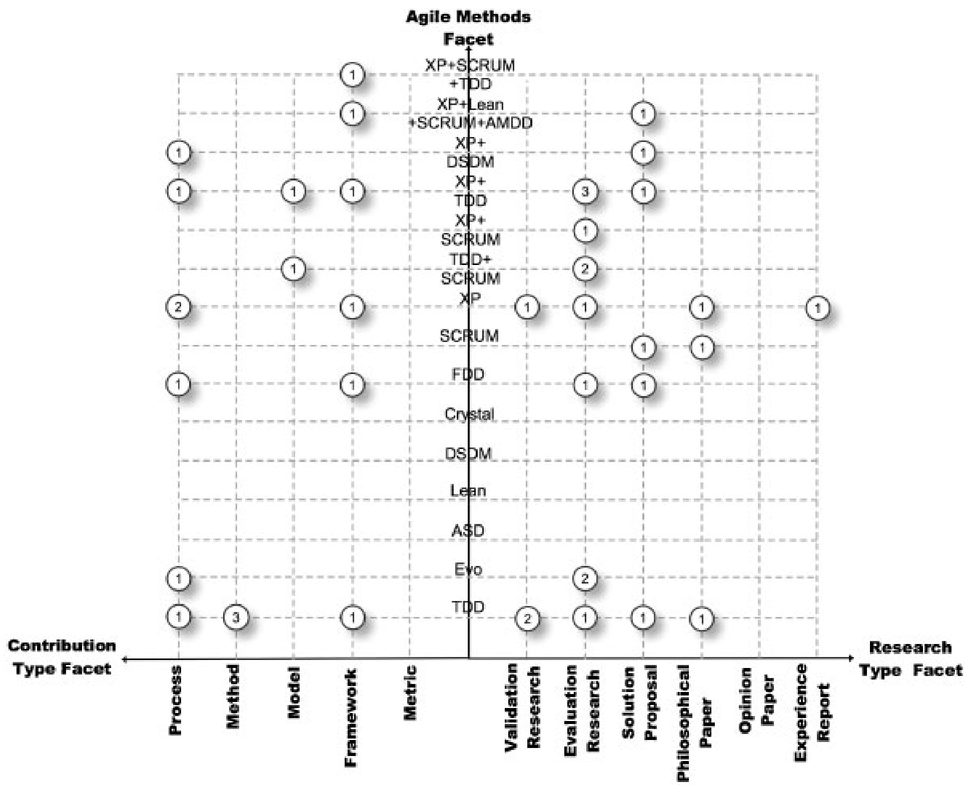

Figure 6. Contribution type versus agile methods versus Research Type.

3.1.3. RQ3—SEI's PLPF and agile methods. The PLPF is considered a mature [1] and established source on SPL practice areas. Some of the primary studies [6, 12, 20, 24, 36, 38, 39] explicitly address these practice areas. For the other studies, we inferred the practice areas from the SPL activities, artifacts, and roles described in the text of the study.

The *architecture definition*, *component development*, *requirements engineering*, *testing*, *scoping*, *and tool support* practice areas are focused on the software development process where Agile methods have more applicability. Ghanam [15, 18, 19, 30–32] and Riegger [34] both focus on testing, however their work overlaps with the *architecture definition*, *component development*, *requirements engineering*, *and tool support* practice areas. Tourret [26] and Xiaochen [25] apply DSDM/XP and FDD in the *architecture definition*, *component development*, *requirements engineering*, *and scoping* practice areas. Carbon [4] also addresses *architecture definition*, *component development*, *requirements engineering*, *and scoping* practice areas, however, through the use of Agile practices *planning game* and *incremental design*. The studies that address *component development*, also address *architecture definition* [4, 16, 19, 21, 25, 26, 35] as these practice areas are closely related technically. The *tool support* practice area is addressed by Riegger [34] who looks at tool support for *requirements engineering* and *testing*.

Agile practices, such as *pair programming*, *TDD*, *refactoring*, and *collective code*, are most frequently applied in the *Architecture Definition*, *Architecture Evaluation*, *Component Development and Testing* practice areas.

The PLPF areas of *Mining Existing Assets*, *Understanding Relevant Domains*, *Using Externally Available Software*, *Make/Buy/Mine/Commission Analysis*, *and Process Discipline* are organizational practice areas. Agile methods with strong organizational practices such as Lean, DSDM, and ASD could contribute to these practice areas. However, currently, there are not enough empirical studies reporting work in these areas.

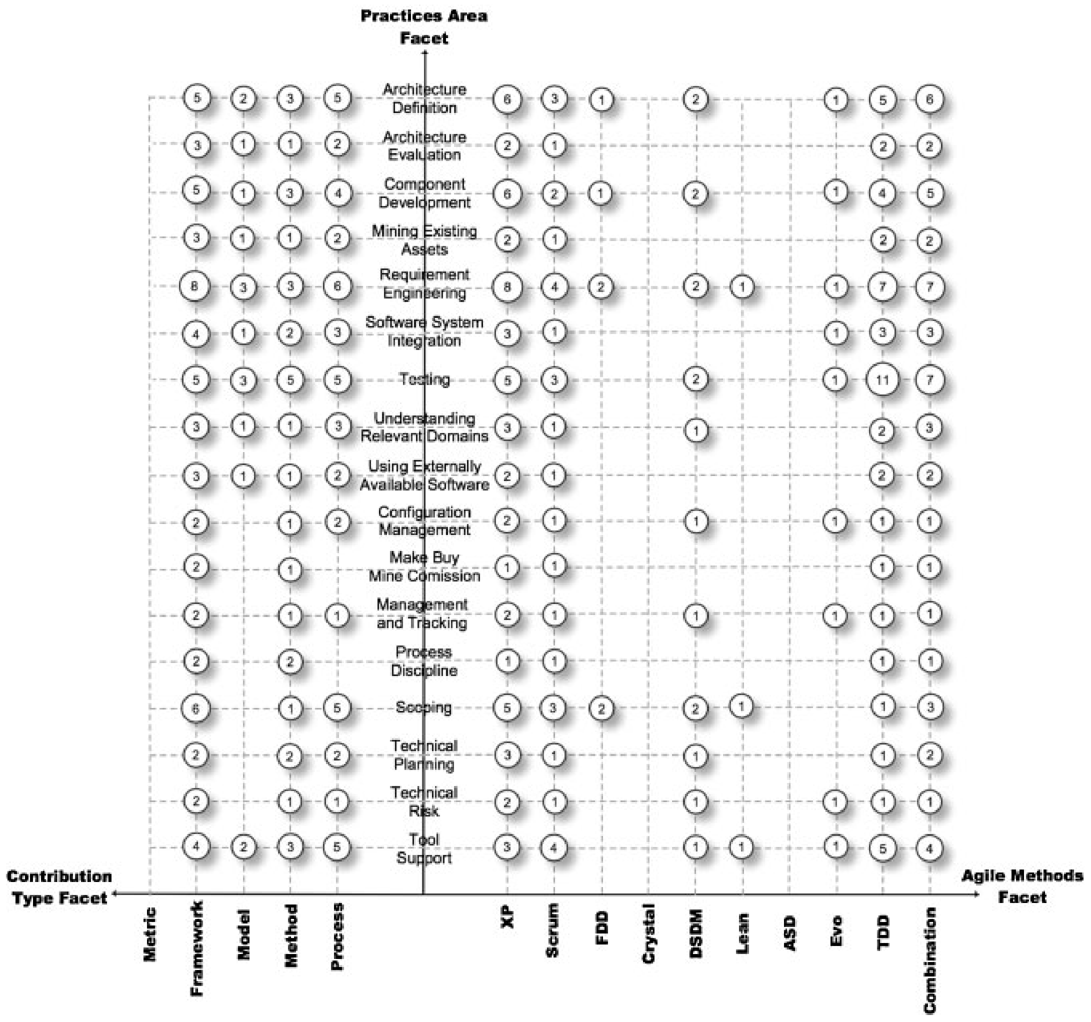

Figure 7. Contribution type versus practice areas versus agile methods.

In Figure 7, the contribution of particular Agile methods is mapped to the PLPF. The most popular Agile methods are XP, SCRUM, and TDD [49]. XP, SCRUM, and TDD are also the most popular in the SPL context. These methods have practices that consider both the development and management aspects of software development, therefore having a larger scope for contributing to the SPL practice areas.

In general, more than one method is used in the same practice area [16, 23], highlighting the importance of combining different Agile methods to meet the needs of a particular SPL practice area.

One important issue is that the Agile approaches identified in this study cover different areas of SPL. No single study considers the whole SPL lifecycle. Full coverage of SPL can only be provided by a merging of current Agile approaches proposing methods or Agile approaches considering frameworks. No Agile studies contributed to metrics in SPL, it could help to capture important aspects regarding the integration and to better understand the impact of the integration in software quality, size/complexity, cost, and so on. Agile frameworks for SPL were the most popular type of studies identified, in particular, Agile frameworks for the *Requirements Engineering* practice area. The *Testing* practice area was the second most popular for adoption of Agile methods. Finally, the suggested Agile approaches were equally spread across the contribution types.

Table V. Number of studies per strategies.

Strategies	References	Number of studies
Pro-active	[4, 16, 17, 23, 25, 28, 33, 35, 37, 38, 42]	11
Reactive	[18–20, 24, 30, 37, 40]	7
Incremental	[20, 29, 27, 37]	4

3.1.4. RQ4—SPL strategies and agile methods. According to McGregor [12], there are three widely recognized strategies (pro-active, reactive, and incremental) that an SPL can pursue with respect to investment in core assets. No particular SPL strategy is the best approach in all cases, and the choice of SPL strategy depends on how the organization operates.

Ghanam [18, 19, 30] and O'Leary [24] describe their Agile SPL approaches as reactive (bottom-up). In these cases, the Agile approaches have a major influence on software development. For example, Ghanam has TDD as the main aspect to guide the building of the software product line.

Kurman [38], Babar [16], Carbon [4, 28], Martinez [23], Feng [17], and Xiachen [25] describe pro-active (up-front SPL) strategies where the Agile approaches are wrapped into frameworks, processes, methods, or models for SPL.

Only Gylterud [20] and Noor [27, 29] describe approaches using the incremental strategy. In [37], the authors discuss some aspects regarding the integration between Agile and SPL, such as the architectural issue, which deals with component dependencies for easy plug-in/plug-out of features. This discussion also considers all SPL strategies. Twelve studies [5, 6, 12, 15, 21, 22, 31, 32, 34, 39, 41, 43] did not identify the SPL strategy adopted. Many of these studies described approaches independent of the SPL strategy being applied. The mapping between SPL strategies and the number of Agile studies that address each strategy is shown in Table V. We note that there is a slight tendency for pro-active strategy.

Reactive development identifies artifacts for reuse and 'harvests' them from SPL core assets [12]. This can become a form of re-engineering components for common usage. There is few literature available on the usage of Agile methods for scoping, identifying, or re-engineering components for usage in an SPL. However, more studies are required to confirm this evidence.

However, it should be noted that the reactive strategy is theoretically more inline with the Agile philosophy. The unpredictable and changing nature of the reactive strategy has a larger scope to allow for the potential benefits of Agile methods.

Considering SPL strategies by contribution types (see Figure 8), a tendency towards research proposing frameworks and processes with the pro-active strategy can be observed. One possible reason for this trend is that the greater managed and planned SPL environment under the pro-active strategy affords greater opportunity for integration of Agile methods. However, further evidence on the relationship between Agile methods and SPL strategies is also required.

3.2. Main findings and discussion

The goal of this mapping study is to identify the primary studies that combine Agile and SPL approaches, investigate whether Agile methods and principles are being adopted in Agile SPL approaches, and identify which practice areas from PLPF as well as SPL strategies are adopting Agile methods. None of the Agile SPL approaches considered the complete SPL development process, rather they suggest the adoption of Agile practices in selected aspects of SPL.

While most of the identified studies are empirical (see Figure 9), few studies provide details on validation. Thus, it is difficult to draw general conclusions. Instead, we performed a quality summary of the primary studies using a narrative summary [50, 51] to aggregate, summarize, and combine findings on Agile SPL.

According to Figure 9 a small number [5, 6, 12, 20, 37, 40] of opinion or theoretical studies that investigated the adoption of Agile principles in SPL and key issues for the integration of Agile

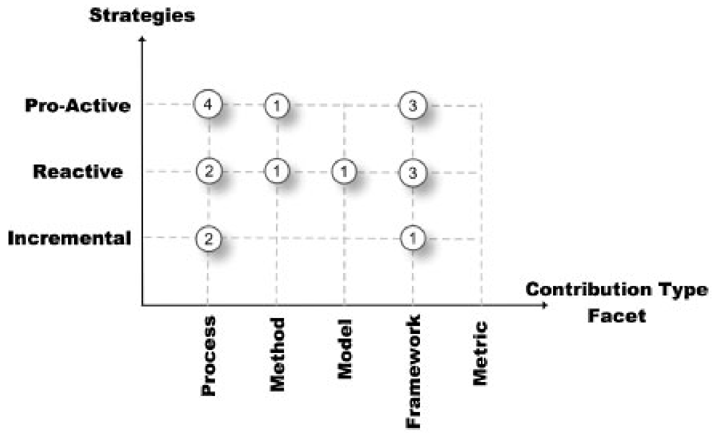

Figure 8. SPL strategies versus contribution type.

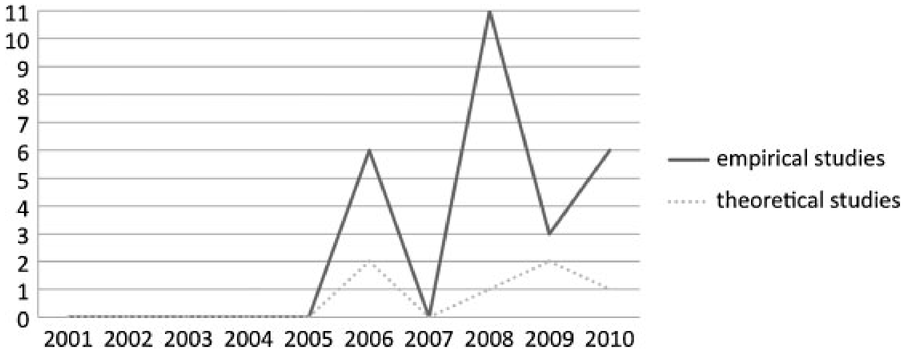

Figure 9. Studies distribution per year.

and SPL were identified. For example, Navarrete [6] proposes an approach to combine Agile and SPL using the CMMI. There are studies about Agile and CMMI, such as [52]. This study shows how the use of Agile practices may support the goals of the CMMI process areas. CMMI and SPL have been explored in [53], this combination may be used as a conceptual framework, bridging the gap between Agile and SPL. In another example, Mohan [40] investigates the fusion of Agile practices with SPL practices through the *Theory of Complex Adaptive System* (*CAS*) [54]. The theory helps explain the use of Agile principles and their integration within an SPL environment.

The main advantage of these theoretical studies is the analysis on combining of practices. However, a drawback on these theoretical studies is the lack of detailed analysis on integration of Agile and SPL practices. For example, there is no systematic analysis on the risks of Agile SPL integration.

3.2.1. Integration between agile manifesto principles and SPL approach. Empirical studies have focused on particular Agile principles, such as the '… *early and continuous delivery of valuable software*.', '… *technical excellence* …' and '*Simplicity* …'. In the first principle, we think that '… *valuable software*.' may be abstracted by '… *valuable artifact*.' within SPL, i.e. artifacts (models, specifications, test documents, etc.) that deliver value to the project. It is desired that both Agile and SPL approaches deliver these artifacts in an *early and continuous* way to the customer.

The '… *technical excellence* …' and '*Simplicity* …' Agile principles have paid much attention because of the TDD use. TDD has been used to manage SPL variability from features and requirements to architecture, components, and testing. However, the validation of TDD has only been conducted in 'toy systems'. It is necessary to validate TDD with empirical evidence and in multiple activities, such as, product derivation, in order to agree with the initial good results in terms of *simplicity* and *technical excellence*.

The '*The best architectures* ...' principle has been addressed by TDD, DSDM, FDD, SCRUM [16] and XP. TDD [18, 19, 30–32], DSDM [26], and FDD [25] have been used as a basis to insert SPL practices or activities. XP practices, *planning game* and *incremental design* [4] have been used in the PulSE-I SPL process. In these situations, we can observe the versatility of combining Agile and SPL. For example, the organizations can adjust their processes according to specific goals (increasing process discipline, interactiveness, incremental, or evolutionary process).

Babar [16] describes an industry case study with an Agile product development approach, however, the adoption of Agile principles has not been discussed in detail. The studies of Xiaochen [25] and Tourret [26] suggest Agile approaches, but it is necessary to evaluate these in an industrial context. These studies discuss the use of Agile principles in domain engineering or application engineering, however, little detail is provided.

Collaborative engineering seems particularly suited to the combining of Agile and SPL. In Noor [27, 29], this combination is explored. Collaborative engineering techniques improve the collaboration among SPL stakeholders—and address the '*Business people and developers must work together* ...' and '... *face-to-face conversation*' Agile principles. Agile practices can also be used in this context, for example, *planning game* and *on-site customer* from XP, and *sprints* from Scrum.

3.2.2. Agile methods being used within the SPL approach. Although Agile methods such as Evo, FDD, DSDM, and TDD have been completely applied in the SPL process, others, such as XP, Scrum, and Lean, have had limited exposure to SPL integration. This phenomenon can be explained in the context of organizational impact. Organizations choose low risk strategies for the adoption of Agile methods, by the inclusion of Agile practices into specific SPL activities with limited scope.

The up-front adoption of an Agile SPL approach can hide or mask the impact on the use of particular Agile practices. In cases where an Agile method has been applied entirely within an SPL environment, the approach was not evaluated [25, 26] in an industrial setting, or the organization had not the maturity to adopt an SPL approach [33].

Despite several studies dealing with XP and SCRUM in SPL, only some Agile practices have been applied (*planning game*, *incremental design*, *on-site customer*, and *pair programming* from XP; *sprint*, *daily meeting*, and *retrospective* from Scrum). For example, if the organization needs to adjust or manage the schedule of the project, it can use *retrospectives* (Scrum) and/or *daily meeting* (Scrum) to improve the communication and feedback among stakeholders. However, Agile experts advise that Agile practices should be applied jointly to allow a synergy benefit among them. A mix-and-match approach to Agile practices might impact the potential benefits.

There was no evidence of the integration of ASD or Crystal in SPL. This could be due to the low impact of these methods in the industry and research, compared to the other more popular methods, such as SCRUM and XP [49]. However, in the same way that FDD and DSDM have been combined with SPL, ASD or Crystal could be applied. ASD has activities such as *project initiation*, that addresses *project mission*, *objectives, and scope constraints* of the project. These activities have a focus similar to the scoping activity in SPL. The methodology of the Crystal family of Agile methods, can be adapted from simple, trivial systems to complex and large systems. The adaptation chosen depends on the number of people in the project, criticality of the system/domain, and priority of the project.

There is evidence related to the combination of Agile methods for integration with SPL. For example, DSDM and XP [26] have been combined where DSDM provides the Agile management practices and XP provides the Agile development practices for use in particular DSDM iterations.

XP and SCRUM [16] have also combined practices for integration with SPL. However, as in the previous studies, there is a lack of empirical evidence of these approaches in the industrial context.

3.2.3. PLPF practices areas addressed by agile methods. The use of Agile methods in *architecture definition*, *component development*, *requirements engineering*, *testing*, *scoping*, *and tool support* PLPF practice areas has been proposed. In fact, the integration of Agile practices is most often proposed within the Software Engineering and Technical Management PLPF practice areas. Despite the presence of primary studies that address these practice areas, we do not see these being developed into a single all-encompassing Agile SPL approach. Therefore, further studies are required that consider a single Agile SPL approach which considers all the PLPF areas. In addition, such studies might provide evidence of synergy benefits, which do not exist in selected adoption of Agile practices.

The SPL *scoping* activity has been addressed by XP, Scrum, FDD, DSDM methods, and Collaboration Engineering practices. Agile methods do not consider domain scoping as in SPL, however, they have applicable practices, such as *planning game*, *feasibility study*, and *business study*.

The use of Agile practices in SPL *requirements engineering* area has been explored with XP, Scrum, FDD, DSDM, and Collaboration Engineering. Agile methods, traditionally, have lightweight processes for *requirements engineering*. The main challenge is balancing Agile values and principles and SPL artifacts, activities, and roles in the requirements engineering processes. The purpose is to streamline the process, keeping important artifacts for the later phases.

Architecture definition and *component development* have been addressed by XP (*design simple*, *refactoring*, *continuous integration*), FDD (*design by feature*, *build by feature*), DSDM (*system architecture definition*), and TDD (*acceptance*, *unit tests*) practices. Even with some studies addressing these SPL practice areas, we need to investigate in detail the balancing required for some architectural attributes, such as flexibility and incrementally. In addition, refactoring needs to be investigated in the pro-active, incremental, and reactive reuse context because new issues related to design debt [55] may emerge from the combination between Agile and SPL approaches.

Testing has been addressed by XP and TDD practices. These Agile practices are adopted to avoid extra work (*Simplicity* Agile principle), embrace concerns with artifact quality (*... excellence technical* Agile principle), and streamline the design, implementation, and test activities.

3.2.4. SPL strategies commonly adopted to introduce agile methods in SPL. Twelve studies did not clearly identify the adopted SPL strategy in which Agile integration was applied, however, they proposed approaches that could apply to all strategies. The majority of the studies have indicated the used strategy while combining Agile practices and SPL. The number of studies that suggested Agile practices in a reactive strategy was influenced by research into SPL adoption of TDD. TDD has been applied to extract core assets in a bottom-up way from the SPL products. For the incremental strategy, the re-engineering of legacy-products to introduce SPL is proposed [29]. The process focuses on *scoping*, however, it does not explore other practice areas from SPL, such as *requirements engineering*, *architecture definition*, and *testing*.

We cannot say which specific strategy facilitates Agile integration, as the studies showed Agile practices, principles, and methods being applied in all strategies. However, for Agile methods the reactive SPL strategy does not mean to 'react' to each product as new and unique. The reactive strategy is more adaptive to changes in the business climate or in the domain and product definitions than the pro-active strategy. Thus, we believe that reactive strategy is more suitable to Agile approaches.

3.2.5. General discussion. As variability is an important aspect from SPL, research into the impact of Agile practices needs to be conducted. In other words, how can Agile practices address SPL

Table VI. Number of studies per research questions.

Research questions	Studies	Number of studies
RQ1	[4, 6, 12, 15–35]	24
RQ2	[16, 4, 28, 17, 18, 31, 30, 15, 19–21, 33, 35, 38, 23–25, 41, 34, 5, 26, 43]	23
RQ3	[4–6, 12, 15–35, 37–43]	32
RQ4	[4, 16–20, 23–25, 27–30, 33, 35, 37, 38, 40, 42]	19

variability? In [15, 18, 19, 30–32, 35], TDD is proposed as a way to handle variability. These preliminary results on the use of TDD to address SPL variability is a promising area, since through this combination, we may address traceability among the system artifacts, such as code, requirements, variation point, variants, features, and architecture. In addition, automated acceptance tests may replace or minimize traditional documentation for SPL purposes.

There is a need to investigate the integration of Agile practices in other SPL aspects, such as the application of variability mechanisms (for example: inheritance, parameterization, and design/analysis patterns) in different abstraction levels (feature models, requirements, design, code, tests) when using Agile practices. For instance, the *planning game* Agile practice from XP could be adopted to include notes or fields into the story card in order to represent variations and variation points.

It is necessary further evaluation of the associated risks and challenges of the integration of Agile and SPL practices. For example, *refactoring and continuous integration* are important practices in Agile with risks and challenges, such as design debt [55]. Nevertheless, combining these Agile practices with SPL practices, such as *architecture definition and component development* can bring other risks and challenges.

There is a lack of experienced papers on Agile SPL. Papers reporting on the experiences of implementing techniques and tools in the industry are desirable. These reports are important sources for improving the evidence on a topic. Although [20] has described some evidence, it is necessary to also collect further evidences, for example, from industrial experts about Agile SPL.

There is no study on the use metrics and this is a gap in the literature that needs to be filled. Studies on the use of metrics in Agile SPL should be considered for further investigations to quantify to what degree the adoption of Agile practices in SPL is appropriate.

Finally, there were a number of studies that did not explicitly answer the research questions set out in this study. In Table VI, the mapping between the studies and the research question is presented. Even with some studies addressing the research questions, more evidence is required to prove the effectiveness of the integration of Agile practices in SPL in terms of productivity, risks, challenges, cost, and so on. This topic—Agile Software Product Line—is still an emerging research area being evaluated and validated in the industry and academia, however, further investigation in the field is required.

4. THREATS TO VALIDITY

There are some threats to the validity of our study, which we briefly describe along with the mitigation strategy for each threat.

- *Research questions*: The research questions we defined cannot provide complete coverage of the Agile SPL field. We had considered this as a feasible threat; however, we had several discussions with project members and Agile SPL experts to validate the questions. In this way, although we had not selected the optimum set of questions, we attempted to address the most frequently asked and those questions that considered open issues in the field.
- *Publication bias*: We cannot guarantee that all relevant primary studies were selected. It is possible that some relevant studies were not chosen during the search process. We mitigated this threat as much as possible, by following references in the primary studies.

- *Search conducted*: The digital databases do not have compatible search rules. We adapted our search strings for each digital database. Nevertheless, we did not know all the rules that digital database used to search a document. We mitigated this threat by running the search in 13 digital databases.
- *Data extraction*: During the extraction process, the studies were classified based on our judgment. However, despite double checking, some studies could have been classified incorrectly. In order to mitigate this threat, the classification process was performed by more than one researcher.
- *Contribution type facet*: This mapping also showed that the primary studies used the terms: approach, framework, process, model, and method. However, an absence of general usage rules for these terms was observed. To mitigate this risk, we adopted the original authors nomenclature, the same idea applied on [56].
- *Principles deduction*: When analyzing the research questions regarding to the integration of Agile principles and SPL approach. In most cases, the principle being applied was deduced from the Agile practices identified. In order to mitigate this issue, this deduction was verified by two researchers.

5. CONCLUDING REMARKS AND FUTURE WORK

In this paper we presented a systematic mapping study on the adoption of Agile methods in SPL. This study represents the initial evidence in the field.

Initially, the mapping study showed that the primary studies identified themselves using the terms: approach, framework, process, model, and method without clearly defining these terms. The indiscriminate use of these terms can make future analysis difficult.

Another finding was that many studies did not describe the principles used to combine Agile methods with SPL. However, from the Agile practices applied it was possible to identify the underlying principles.

Many studies explored the Agile practices for the most popular Agile methods: XP and SCRUM. The practices from these methods were applied in scoping, requirements engineering, architecture definition, component development, and testing SPL practice areas. The combination of XP and SCRUM prevailed in its application for the pro-active SPL strategy.

DSDM, Lean, FDD, and Evo have been addressed by studies, as well as a combination of these methods, for example, DSDM and XP. We think that there is a large scope for the possibilities of further combinations.

On the other hand, even with some results, many research questions in Agile SPL remain unanswered. There is no complete Agile SPL approach that addresses all the phases of domain engineering, for example. There are no studies treating the risks of each combination between Agile practice and SPL practice. In addition, the studies require further evaluation, mainly in industrial scenarios.

For the future work, the results of this paper will serve to a survey with Agile and Software Product Lines experts to improve the understanding in the area. In addition, we plan to investigate the integration in SPL of more disciplined Agile methods such as Feature-Driven Development, EVO, and Dynamic System Development Management. We believe that these Agile methods are suited to SPL because they provide practices to support important aspects, such as defining the business domain, feature modeling, reuse of artifacts, and process definition.

APPENDIX A: SEARCH ENGINES

See Table AI for digital databases.

Table AI. Digital databases.

Digital databases
ACM Digital Library
Citeseer Library
Elsevier—Engineering Village—Compendex
Google Scholar
IEEE Computer Society Digital Library
IEEEXplore
INSPEC
ISI Web of Knowledge
ScienceDirect
Scirus
Scopus
SpringerLink
Wiley Inter Science Journal Finder (computer science area)

APPENDIX B: JOURNALS AND CONFERENCES

See Tables BI–BII for Journals and Conferences.

Table BI. Journals.

Journals list
ACM Computing Survey
ACM Transactions on Software Engineering and Methodology (TOSEM)
Annals of Software Engineering
Communications of the ACM (CACM)
Empirical Software Engineering Journal
IEEE Software
IEEE Transactions on Software Engineering
IET Software
Information and Software Technology
Journal of Object Technology (JOT)
Management Science
Software Practice and Experience (SPE) Journal
Software Process Improvement and Practice

Table BII. Conferences.

Conferences list
Agile Business Conference
Agile Development Conference (ADC)
Asia-Pacific Software Engineering Conference (APSEC)
Australian Software Engineering Conference (ASWEC)
Automated Software Engineering (ASE)
Computer Software and Applications Conference (COMPSAC)
Empirical Software Engineering and Measurement (ESEM)
Euromicro Conference—Software Engineering and Advanced Applications (SEAA)
European Conference for Object-oriented Programming (ECOOP)
European Conference on Software Architecture (ECSA)
European Software Engineering Conference (ESEC)
Fundamental Approaches to Software Engineering (FASE)
Generative Programming and Component Engineering (GPCE)
International Conference and Workshop on the Engineering of Computer-based Systems (ECBS)
International Conference on Advanced Information Systems Engineering (CAiSE)
International Conference on Enterprise Information Systems (ICEIS)

Table BII. *Continued.*

International Conference on Information Reuse and Integration (IRI)
International Conference on Product Focused Software Development and Process Improvement (PROFES)
International Conference on Program Comprehension (ICPC) International
Conference on Quality Software (QSIC)
International Conference on Software Engineering (ICSE)
International Conference on Software Engineering and Knowledge Engineering (SEKE)
International Conference on Software Process (ICSP)
International Conference on Software Reuse (ICSR)
International Conference on the Quality of Software Architectures (QoSA)
International Symposium on Component-based Software Engineering (CBSE)
Object-oriented Programming, Systems, Languages, and Applications (OOPSLA)
Software Product Line Conference (SPLC)
Software Product Family Engineering (PFE)
Engineering of Computer-based Systems (ECBS)
Variability Modeling of Software-intensive Systems (VaMoS)
Working IEEE/IFIP Conference on Software Architecture (WICSA)
International Conference on Agile Processes and Extreme Programming in Software Engineering

REFERENCES

1. Clements P, Northrop L. *Software Product Lines*: *Practices and Patterns*. Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, U.S.A., 2001.
2. Cockburn A. *Agile Software Development*. Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, U.S.A., 2001.
3. Beck K, Beedle M, van Bennekum A, Cockburn A, Cunningham W, Fowler M, Grenning J, Highsmith J, Hunt A, Jeffries R, Kern J, Marick B, Martin RC, Mellor S, Schwaber K, Sutherland J, Thomas D. Manifesto for agile software development, 2001. Available at: http://agilemanifesto.org/ [7 July 2010].
4. Carbon R, Lindvall M, Muthig D, Costa P. Integrating product line engineering and agile methods: Flexible design up-front vs. incremental design. *APLE '06*: *Proceedings of the First International Workshop on Agile Product Line Engineering in Conjunction with SPLC 2006*, Baltimore, Maryland, U.S.A., 2006.
5. Tian K, Cooper K. Agile and software product line methods: Are they so different? *APLE '06*: *Proceedings of the First International Workshop on Agile Product Line Engineering in Conjunction with SPLC 2006*, Baltimore, Maryland, U.S.A., 2006.
6. Navarrete F, Botella P, Franch X. An approach to reconcile the agile and cmmi contexts in product line development. *APLE '06*: *Proceedings of the First International Workshop on Agile Product Line Engineering in Conjunction with SPLC 2006*, Baltimore, Maryland, U.S.A., 2006.
7. Linden FJvd, Schmid K, Rommes E. *Software Product Lines in Action*: *The Best Industrial Practice in Product Line Engineering*. Springer New York, Inc.: Secaucus, NJ, U.S.A., 2007.
8. Kitchenham B, Charters S. Guidelines for performing systematic literature reviews in software engineering. *Technical Report EBSE 2007-001*, Keele University and Durham University Joint Report, 2007.
9. Petersen K, Feldt R, Mujtaba S, Mattsson M. Systematic mapping studies in software engineering. *EASE '08*: *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, University of Bari, Italy, 2008.
10. Brereton P, Kitchenham BA, Budgen D, Turner M, Khalil M. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software* 2007; **80**(4):571–583.
11. Budgen D, Turner M, Brereton P, Kitchenham B. Using mapping studies in software engineering. *PPIG'08*: *In 20th Annual Meeting of the Psychology of Programming Interest Group*, Lancaster University, 2008; 195–204.
12. McGregor J. Agile software product lines, deconstructed. *Journal of Object Technology* 2008; **7**(8):7–19.
13. Kitchenham B, Mendes E, Travassos G. Cross versus within-company cost estimation studies: A systematic review. *IEEE Transactions on Software Engineering* 2007; **33**(5):316–329.
14. Wieringa R, Maiden N, Mead N, Rolland C. Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. *Requirements Engineering* 2005; **11**(1):102–107.
15. Ghanam Y, Maurer F. Linking feature models to code artifacts using executable acceptance tests. *SPLC '10*: *Proceeding of Software Product Line Conference*, Jeju Island, South Korea, 2010.
16. Babar MA, Ihme T, Pikkarainen M. An industrial case of exploiting product line architectures in agile software development. *SPLC '09*: *Proceedings of the 13th International Software Product Line Conference*, San Francisco, CA, 2009; 171–179.
17. Feng K. Towards a knowledge based agile product line requirements engineering framework: Collection of expertise and its application to RUP based process templates. *PhD Thesis*, University of Texas, Dallas, U.S.A., 2008.

18. Ghanam Y, Maurer F. An iterative model for agile product line engineering. *The SPLC Doctoral Symposium, 2008—in Conjunction with the SPLC '08*: *Proceedings of the 12th International Software Product Line Conference*, Limerick, Ireland, 2008.
19. Ghanam Y, Andreychuk D, Maurer F. Reactive variability management using agile software development. *Agile '10*: *Proceedings of the International Conference on Agile Methods in Software Development*, Nashville, U.S.A., 2010.
20. Gylterud S. Practices of agile software product-line engineering: A qualitative assessment of empirical studies. *Master's Thesis*, Norwegian University of Science and Technology, Norway, 2009.
21. Hanssen G, Yamashita AF, Conradi R, Moonen L. Software entropy in agile product evolution. *HICSS '10*: *Proceedings of the 2010 43rd Hawaii International Conference on System Sciences*. IEEE Computer Society: Washington, DC, U.S.A., 2010; 1–10.
22. Karam M, Dascalu S, Safa H, Santina R, Koteich Z. A product-line architecture for web service-based visual composition of web applications. *Journal of Systems and Software* 2008; **81**(6):855–867.
23. Martinez J, Diaz J, Perez J, Garbajosa J. Software product line engineering approach for enhancing agile methodologies. *Lecture Notes in Business Information Processing—Agile Processes in Software Engineering and Extreme Programming*, vol. 31. Springer: Berlin, Heidelberg, 2009; 247–248.
24. O'Leary P. Towards a product derivation process reference model for software product line organizations. *PhD Thesis*, University of Limerick, Limerick, Ireland, 2010.
25. Paige RF, Wang X, Stephenson ZR, Brooke PJ, Towards an agile process for building software product lines. *Lecture Notes in Computing Science—Extreme Programming and Agile Processes in Software Engineering* 2006; **4044**:198–199.
26. Tourret R. Using agile methods to develop software product lines. *Master's Thesis*, Computer Science—University of York, York, England, 2006.
27. Noor MA, Rabiser R, Grünbacher P. Agile product line planning: A collaborative approach and a case study. *Journal of Systems and Software* 2008; **81**(6):868–882.
28. Carbon R, Knodel J, Muthig D, Meier G. Providing feedback from application to family engineering—The product line planning game at the Testo AG. *SPLC '08*: *Proceedings of the 2008 12th International Software Product Line Conference*, Limerick, Ireland, 2008; 180–189.
29. Noor MA, Rabiser R, Grünbacher P. A collaborative approach for reengineering-based product line scoping. *APLE '06*: *Proceedings of the First International Workshop on Agile Product Line Engineering in Conjunction with SPLC 2006*. Baltimore, Maryland, U.S.A., 2006.
30. Ghanam Y, Maurer F. Extreme product line engineering refactoring for variability: A test-driven approach. *XP '10*: *11th International Conference on Agile Processes and eXtreme Programming*, Trondheim, Norway, 2010.
31. Ghanam Y, Maurer F. Extreme product line engineering: Managing variability and traceability via executable specifications. *AGILE '09*: *Proceedings of the 2009 Agile Conference*, Chicago, U.S.A., 2009; 41–48.
32. Ghanam Y, Park S, Maurer F. A test-driven approach to establishing & managing agile product lines. *The 5th Software Product Lines Testing Workshop* (*SPLiT 2008*) *in Conjunction with SPLC 2008*, Limerick, Ireland, 2008; 151–156.
33. Hanssen GK, Fígri TE. Process fusion: An industrial case study on agile software product line engineering. *Journal of Systems and Software* 2008; **81**(6):843–854.
34. Riegger F. Test-based feature management for agile product lines. *Master's Thesis*, Agile Software Engineering Group Department of Computer Science University of Calgary, Calgary, Canada, 2010.
35. Kakarontzas G, Stamelos I, Katsaros P. Product line variability with elastic components and test-driven development. *CIMCA '08*: *Proceedings of the 2008 International Conference on Computational Intelligence for Modelling Control & Automation*. IEEE Computer Society: Washington, DC, U.S.A., 2008; 146–151.
36. O'Leary P, McCaffery F, Thiel S, Richardson I. An agile process model for product derivation in software product line engineering. *Journal of Software Maintenance and Evolution*: *Research and Practice* 2010; DOI: 10.1002/smr.498.
37. Ghanam Y, Maurer F, Abrahamsson P, Cooper K. A report on the XP workshop on agile product line engineering. *SIGSOFT Software Engineering Notes* 2009; **34**(5):25–27.
38. Kurmann R. Agile spl scm agile software product line configuration and release management. *APLE '06*: *Proceedings of the First International Workshop on Agile Product Line Engineering in Conjunction with SPLC 2006*, Baltimore, Maryland, U.S.A., 2006.
39. McGregor J. Mix and match. *Journal of Object Technology* 2008; **7**(6):7–16.
40. Mohan K, Ramesh B, Sugumaran V. Integrating software product line engineering and agile development. *IEEE Software* 2010; **27**(3):48–55.
41. Raatikainen M, MyllŁrniemi V, MŁnnist T. Towards managing development by analyzing integration of backlog and feature model. *NW-MoDE '08*: *Nordic Workshop on Model Driven Engineering*, Reykjavik, Iceland, 2008; 247–248.
42. Trinidad P, Benavides D, Ruiz-Cortés A, Segura S, Toro M. Explanations for agile feature models. *APLE '06*: *Proceedings of the First International Workshop on Agile Product Line Engineering in Conjunction with SPLC 2006*, Baltimore, Maryland, U.S.A., 2006.
43. Trinidad P, Benavides D, Durán A, Ruiz-Cortés A, Toro M. Automated error analysis for the agilization of feature modeling. *Journal of Systems and Software* 2008; **81**(6):883–896.

44. Larman C. *Agile and Iterative Development*. Addison-Wesley Professional: Sardina, Italy, 2003.
45. Ambler SW. *The Object Primer*: *Agile Model-Driven Development with UML 2.0*. Cambridge University Press: New York, U.S.A., 2004.
46. Atkinson C, Kuhne T. Model-driven development: A metamodeling foundation. *IEEE Software* 2003; **20**(5):36–41.
47. Boehm B, Turner R. *Balancing Agility and Discipline—A Guide for the Perplexed*. Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, U.S.A., 2003.
48. Read K. Supporting agile teams of teams via test driven design. *Master's Thesis*, Agile Software Engineering Group Department of Computer Science University of Calgary, Calgary, Canada, 2005.
49. Tore D, Torgeir D. Empirical studies of agile software development: A systematic review. *Information Software Technology* 2008; **50**(9–10):833–859.
50. Cruzes DS, Dyba T. Synthesizing evidence in software engineering research. *Empirical Software Engineering and Measurement—ESEM'10*, Bolzano-Bozen, Italy, 2010.
51. Dixon-Woods M, Agarwal S, Jones D, Young B, Alex S. Synthesising qualitative and quantitative evidence: A review of possible methods. *Journal of Health services Research and Policy* 2005; **10**(1):45–53.
52. Pikkarainen M. Towards a better understanding of cmmi and agile integration—Multiple case study of four companies. *Product-focused Software Process Improvement* (*Lecture Notes in Business Information Processing*, vol. 32). Springer: Berlin, Heidelberg, 2009; 401–415.
53. Jones LG, Northrop LM. Product line adoption in a cmmi environment. *Technical Report CMU/SEI-2005-TN-028*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, U.S.A., 2005.
54. Miller JH, Page SE. *Complex Adaptive Systems*: *An Introduction to Computational Models of Social Life*. Princeton University Press: Princeton, New Jersey, U.S.A., 2007.
55. Neill CJ, Laplante PA. Paying down design debt with strategic refactoring. *Computer* 2006; **39**(12):131–134.
56. Zelkowitz MV, Wallace DR. Experimental models for validating technology. *Computer* 1998; **31**(5):23–31.