Accepted Manuscript

Agile Product Line Engineering - A Systematic Literature Review
Jessica Díaz, Jennifer Pérez, Pedro P. Alarcón and Juan Garbajosa

DOI: 10.1002/spe.1087
http://dx.doi.org/10.1002/spe.1087
Software: Practice and Experience

Received 15 July 2010; Revised 28 February 2011; Accepted 18 March 2011
Published online 4 May 2011 in Wiley Online Library (wileyonlinelibrary.com).

Please cite this article as:
Volume 41, Issue 8, pages 921–941, July 2011. Article first published online: 4 MAY 2011 DOI:
10.1002/spe.1087 Wiley InterScience (2011)

Copyright © Wiley-Blackwell

This is the PDF file of the manuscript that was accepted for publication. The manuscript
have undergone copyediting, typesetting, and review of the resulting proof in its final form.
Please note that during the production process, errors may had been discovered which
may have affected the content, and all legal disclaimers that apply to the journal pertain.
Agile Product Line Engineering - A Systematic Literature Review

Jessica Díaz, Jennifer Pérez, Pedro P. Alarcón and Juan Garbajosa

Universidad Politécnica de Madrid - Technical University of Madrid (UPM) E.U. Informática Cra. Valencia Km. 7 E-28031 Madrid, Spain yesica.diaz -at- upm.es, {jenifer.perez,pedrop.alarcon,jgs} -at- eui.upm.es

SUMMARY

Software product line engineering (SPLE) demands upfront long-term investment in (i) designing a common set of core-assets and (ii) managing variability across the products from the same family. When anticipated changes in these core-assets have been predicted with certain accuracy, SPLE has proved significant improvements. However, when large/complex product-line projects have to deal with changing market conditions, alternatives to supplement SPLE are required. Agile software development (ASD) may be an alternative, as agile processes harness change for the customer’s competitive advantage. However, when the aim is to scale agile projects up to effectively manage reusability and variability across the products from the same family, alternatives to supplement agility are also required. As a result, a new approach called agile product line engineering (APLE) advocates integrating SPLE and ASD with the aim of addressing these gaps. APLE is an emerging approach, what implies that organizations have to face with several barriers to achieve its adoption. This paper presents a systematic literature review of experiences and practices on APLE, in which the key findings uncover important challenges about how to integrate the SPLE model with an agile iterative approach to fully put APLE into practice.

1. INTRODUCTION

Software product line engineering (SPLE) [1, 2] exploits the commonality and variability across products from the same family by an upfront long-term design of a common set of core-assets/platform (Domain Engineering, DE); then core-assets are assembled into customer-specific products by deriving the existing variability across the products (Application Engineering, AE). This means that SPLE takes advantage of common features among the products of a family through the systematic reuse of the core-assets and the effective management of variabilities across the products. Significant improvements have been made using SPLE, such as reducing cost and time-to-market, and providing flexibility to respond to planned changes [3] in contexts in which anticipated changes can be predicted with certain accuracy. However, when large/complex Software Product Lines (SPL) projects have to deal with changing market conditions, i.e. unplanned changes, alternatives to supplement SPLE are required [4]. Agile software development (ASD) may be one of these alternatives, since agile processes harness change for the customer’s competitive advantage [5]. However, scalability is still a challenge in agile projects; hence reusability and variability management among the products from the same family are not easily scaled up [6]. As a consequence, an approach resulting from the integration of SPLE and ASD, complementing each other, might make sense. Some advocates of this approach have coined the term agile product line engineering (APLE) [7]; others coined the term agile software product lines (ASPL) [4] to refer to the same approach.

Although difficult to provide a date, it can be said that the need to consider SPLE and ASD as a joint approach was recognized in 2002. In conjunction with the 2002 International Conference...
on Software Reuse, a new workshop on Software Reuse and Agile Approaches was set up [8]. This workshop focused on combining DE and Agile methods. Later on, in conjunction with the 2006 SPL Conference, a new workshop called ‘Agile Product Line Engineering’ (APLE’06 [7]) was arranged. In this workshop, it was acknowledged that both Agile and SPL approaches share several common goals and could work together. According to the panel ‘Agile and Product Line Engineering’ (APLE’08 [9]), organized by the 2008 SPL Conference, the applicability of agile methods to AE looked like feasible but it was not clear how agile methods could be applied to DE. Finally, two workshops on APLE [10, 11] in conjunction with the eXtreme Programming Conference, were held in 2009 and 2010 respectively. In these workshops, issues, such as the traditional way of creating and managing SPLs, were analyzed, to deal with how the upfront investments in SPL platforms might be profitable in changing market conditions.

Although practitioners and the academic community have tackled the integration of SPLE and ASD from various points of view, no definitive approach has been achieved so far. Organizations still have to face numerous barriers to put APLE into practice. Until now no systematic literature review has been undertaken to bring together the published work on APLE. Systematic literature reviews identify, evaluate and interpret all available relevant research on a specific research question or topic area by using a rigorous and auditable methodology [12]. Systematic literature reviews are important for different reasons [13]: (i) to summarize existing evidence concerning a practice or technology, (ii) to identify where there are gaps in current research, (iii) to help position new research activities; and (vi) to examine how far a given hypothesis is supported or contradicted by the available empirical evidence. As a result, a systematic literature review on APLE could be interesting to build this common understanding and to identify existing challenges to the APLE implementation. In addition, it could be useful for practitioners who want to stay up to date with the state of research and which problems have been solved, as well as for researchers who want to identify where research is lacking.

This paper presents a systematic literature review of experiences and practices on APLE to identify what barriers have been dealt with, and what challenges have to be addressed in the near future to apply APLE to the software industry. With these purposes, a review on the state of the art in APLE has been conducted following the guidelines proposed by Kitchenham [12]. The search strategy identified over 370 unduplicated scientific papers, of which only 39 were directly related to APLE. The key findings uncover important challenges about how to integrate the ‘domain-then-application’ SPLE model [14] with an agile iterative approach.

The remainder of this paper is structured as follows. Section 2 provides an overview of Software Product Line Engineering and Agile Software Development. Section 3 describes the method used for the literature review of the state of the art in APLE. Section 4 reports the contributions of the collected studies. Section 5 summarizes key findings as well as implications for practitioners. Finally, conclusions are presented in Section 6.

2. AN OVERVIEW OF SPLE AND ASD

2.1. Software Product Line Engineering

The literature has established two essential phases to successfully develop SPLs: Domain Engineering and Application Engineering [2]. These phases are described below.

DE consists of creating a set of reusable assets for building systems in a specific problem domain. DE determines the scope of the SPL and handles the commonalities and the variability, i.e. defines the core-assets, points of variations and expected variants for all the products of the SPL. Once the domain has been well understood, an architecture is designed. The Product-Line Architecture is considered flexible in the sense that it is capable of accommodating potential members of the SPL and, proactively, addressing the variability. Reusable assets are implemented and tested, resulting in

*Note of the authors: for the sake of consistency, Domain Analysis, Domain Engineering, and Core Assets Development, are considered to refer to the same phase of SPLE
what is known as a common platform or core-asset base. DE is summarized in the following set of activities or practices [1, 2, 15]:

1. Domain Identification and Scoping,
2. Requirements Engineering,
3. Feature Modeling,
4. Architecture Design,
5. Core Assets Development,
6. Traceability Management,
7. Evolution and Maintenance.

AE† consists of developing products through systematic reuse of core-assets by deriving the SPL variability points, i.e. reusable assets are extended from variability points with the selected variants for a specific product. AE is summarized in the following set of activities and practices [1, 2, 16]:

1. Release Planning (plan for product applications),
2. Product Configuration (model application, architecture application, platform application),
3. Product Development,
4. Test, Integration and Deployment,
5. Traceability Management,

2.2. Agile Software Development

Agility is just an umbrella term for a variety of agile methods that are based on the Agile Manifesto‡. These agile methods are defined in terms of values, principles and practices. Following Shore et al. [5], values are ideals, principles are the application of these ideals to the industry, and practices are principles applied to a specific type of project. Scrum, eXtreme Programming (XP), and Lean Development can be considered under the category of ASD methods. All of them implement iterative incremental life cycles and share some common values and principles and each one of them defines their own practices. Agile methods stress the importance of delivering value to the customer through early and continuous delivery of valuable software. Some of the agile principles are: continuous attention to technical excellence; the best architectures, requirements and designs emerge from self-organizing teams; agile processes harness change for the customer’s competitive advantage; or working software is the primary measure of progress. ASD pursues flexibility in the sense that it values responding to change over following a plan.

3. SYSTEMATIC LITERATURE REVIEW: THE METHOD

A review on the state of the art in APLE has been conducted following the guidelines put forward by Kitchenham [12], who proposes three main phases for doing a systematic review process: (i) planning the review, which aims to develop a review protocol (see Section 3.1); (ii) conducting the review, which executes the planned protocol in the previous phase (see Section 3.2); and (iii) reporting the review, which is responsible for relating the review steps to the community (see Section 4). The execution of the overall process involves iteration, feedback, and refinement of the defined process.

3.1. First Phase: Planning the Review

The phase planning the review consists of developing a review protocol. The review protocol defines the methods to undertake a specific systematic review reducing the possibility that this review can

†Note of the authors: for the sake of consistency, Product Derivation, Application Engineering, and Product Development, are considered to refer to the same phase of SPL e
‡The Agile Manifesto. www.agilemanifesto.org

(2011)
be driven by research expectations. Protocol development must specify (i) the review objective and research questions; (ii) the search strategy; (iii) the explicit inclusion and exclusion criteria; (iv) the criteria to evaluate each study; and (v) the data extraction strategy and the strategy for synthesizing extracted data [12]. All these steps are described in the following subsections.

3.1.1. Review objective and research questions. The review objective is to identify experiences and practices in APLE in order to identify the current APLE approaches, i.e. how SPLE and ASD can be integrated, what barriers have been dealt with, and what challenges have to be addressed in the near future to apply APLE to the software industry. In detail, the research questions that were identified to achieve the outlined objective are the following:

- RQ1: What are the reasons for combining SPLE and ASD? When is it advantageous to put APLE into practice?
- RQ2: How do the principles of SPLE and ASD match with each other?
- RQ3: How are SPLE and ASD positioned with respect to business strategic objectives?
- RQ4: Which current approaches combine SPLE and ASD satisfying AE activities?
- RQ5: What are the challenges and gaps in current approaches that combine SPLE and ASD during AE activities?
- RQ6: Which current approaches combine SPLE and ASD satisfying DE activities?
- RQ7: What are the challenges and gaps in current approaches that combine SPLE and ASD during DE activities?
- RQ8: Which current approaches combine SPLE and ASD satisfying both DE and AE activities through agile principles?
- RQ9: Are there successful industrial experiences putting APLE into practice?

3.1.2. Search strategy. A formal search strategy is required to find the entire population of scientific papers that may be relevant to answer the identified research questions. The formal definition of this search strategy allows us to make a replicable and open review of external assessments. The search strategy consists in defining the search space: electronic databases and conference proceedings that are considered key spaces for the review objective (see Table I). The searches into this space retrieve a list of scientific papers called primary studies. But our search strategy also defines two more steps: secondary searches based on references found in primary studies, and tertiary searches in DBLP based on the authors found in primary studies, to look for previous or subsequent studies. In addition, some authors were directly e-mailed to find out whether they had any material in press.

Table I. Search resources.

<table>
<thead>
<tr>
<th>Data Source</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electronic databases</td>
<td>ACM Digital library</td>
</tr>
<tr>
<td></td>
<td>IEEE Xplore</td>
</tr>
<tr>
<td></td>
<td>SpringerLink</td>
</tr>
<tr>
<td></td>
<td>EI Compendex</td>
</tr>
<tr>
<td></td>
<td>Inspec</td>
</tr>
<tr>
<td></td>
<td>ISI Web of Knowledge</td>
</tr>
<tr>
<td></td>
<td>ScienceDirect</td>
</tr>
<tr>
<td>Conference proceedings</td>
<td>SPLC (Software Product Line Conference)</td>
</tr>
<tr>
<td>Hand searches</td>
<td>XP (Agile Processes and eXtreme Programming in Softw. Eng.)</td>
</tr>
<tr>
<td></td>
<td>Agile Conference</td>
</tr>
<tr>
<td></td>
<td>APLE (Workshop on Agile Software Product Line Engineering)</td>
</tr>
</tbody>
</table>

Once the searching space has been defined, it is necessary to define the search terms to be introduced into the search inquiry forms of electronic databases. The search terms have been divided into two criteria: (i) key words related to ASD and specific agile methods such as Scrum, eXtreme Programming (XP), Feature Driven Development (FDD), or Lean; and (ii) key words related to
SPLE such as product line(s), product family (-ies), Domain Analysis, Domain Engineering, Core Assets, Product Derivation, or Application Engineering. Acronyms for each of these terms were also added in the search inquiry. Next, some examples of searches in EI Compendex and ISI Web of Knowledge respectively are shown:

```
software AND (agile OR agility OR scrum OR “extreme programming” OR fdd OR “feature driven development” OR “feature-driven” OR tdd OR “test driven development” OR “test-driven” OR lean) AND (“product line” OR “product famil”)

Topic=(software) AND Topic=((agile OR agility OR scrum OR “extreme programming” OR fdd OR “feature driven development” OR “feature-driven” OR tdd OR “test driven development” OR “test-driven” OR lean)) AND Topic=((“product line”) OR (“product famil”))
```

3.1.3. **Inclusion and exclusion criteria.** The review protocol also specifies inclusion criteria (IC) and exclusion criteria (EC) which determine whether each potential study should be considered or not for this systematic review. The list of IC and EC for this APLE systematic review is shown below.

- **Inclusion Criteria (IC)**
  - **Type of studies:** Scientific material written in English, according the search terms defined in the previous section. Scientific material is a general term that includes papers, short papers, experience reports, summaries of workshops, panels, and poster sessions, subjected to the normal scientific peer review process, normally double blind reviewed by 2-3 reviewers.
  - **Time:** The scientific material published until June 2010.

- **Exclusion Criteria (EC)**
  - **Agility as adjective:** Those studies that use the term agility as an adjective, but they do not refer to agile methodologies. These studies use the term ‘agile’ or ‘agility’ to mean flexibility but not ASD.
  - **Poor arguments:** Those studies based on general opinion and/or poor arguments.
  - **Reductio ad absurdum:** Those studies that do not fulfill the IC.

3.1.4. **Quality assessment.** Kitchenham’s guidelines [12] suggest performing a quality assessment of each included study; it complements the IC and EC defined above. However, there is no universal agreed definition of “quality”. The Critical Appraisal Skill Programme (CASP)\(^3\) defines criteria for assessing the quality of qualitative research. This systematic review has used the quality criteria defined for CASP and those proposed by Dybå et al. [17]. The criteria cover three main issues: rigor, credibility, and relevance. Briefly, we summarize the quality assessment form defined by [17] in Table II.

<table>
<thead>
<tr>
<th>Quality criteria</th>
<th>YES/NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Is there a clear statement of the aims and objectives of the research?</td>
<td>(YES/NO)</td>
</tr>
<tr>
<td>2. Is there an adequate description of the context in which the research was carried out?</td>
<td>(YES/NO)</td>
</tr>
<tr>
<td>3. Is there an adequate description of the proposed contribution, method, or approach?</td>
<td>(YES/NO)</td>
</tr>
<tr>
<td>4. Is there a clear statement of findings?</td>
<td>(YES/NO)</td>
</tr>
<tr>
<td>5. Is the evidence obtained from experimental studies or observational studies?</td>
<td>(YES/NO)</td>
</tr>
<tr>
<td>6. Is the study of value for research or practice?</td>
<td>(YES/NO)</td>
</tr>
</tbody>
</table>

3.1.5. **Data extraction and data synthesis strategies.** The extraction process consists of identifying the required data to answer the research questions. To this end, we created a set of forms to

---

\(^3\)http://www.phru.nhs.uk/Pages/PHD/CASP.htm

(2011)
store verbatim key concepts regarding aims, findings, and conclusions of each included study. The synthesis process consists of organizing the key concepts to enable comparisons across studies and the reciprocal translation of findings into higher-order interpretations.

3.2. Second Phase: Conducting the review phase

The phase conducting the review includes a sequence of steps: (i) search for primary studies; (ii) selection of primary studies applying inclusion/exclusion criteria; (iii) study quality assessment; (iv) data extraction; and (v) data synthesis.

3.2.1. Search for studies. Following the guidelines described in Section 3.1.2, a search for primary studies was carried out. It retrieved over 536 results, but these results include a certain degree of redundancy between the search engines. Secondary and tertiary searches retrieved 32 citations by scanning reference lists, scanning web pages of prominent authors, and contacting them. This adds up to 568 results. Table III shows the number of studies that were retrieved from each search engine, including duplicated studies. We have rejected 198 duplicated results; hence only 370 unduplicated studies were included to be evaluated in the next step.

Table III. Results of the systematic search for primary studies.

<table>
<thead>
<tr>
<th>Database</th>
<th>Retrieved</th>
<th>Included</th>
<th>Excluded</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM Digital Library</td>
<td>48</td>
<td>4 (8.33%)</td>
<td>44 (91.67%)</td>
</tr>
<tr>
<td>EI Compendex</td>
<td>23</td>
<td>8 (34.78%)</td>
<td>15 (65.22%)</td>
</tr>
<tr>
<td>IEEE Xplore</td>
<td>14</td>
<td>5 (35.71%)</td>
<td>9 (64.29%)</td>
</tr>
<tr>
<td>Inspec</td>
<td>35</td>
<td>9 (25.71%)</td>
<td>26 (74.29%)</td>
</tr>
<tr>
<td>ISI Web of Knowledge</td>
<td>15</td>
<td>6 (40%)</td>
<td>9 (60%)</td>
</tr>
<tr>
<td>ScienceDirect</td>
<td>312</td>
<td>8 (2.56%)</td>
<td>304 (97.44%)</td>
</tr>
<tr>
<td>SpringerLink</td>
<td>89</td>
<td>7 (7.87%)</td>
<td>82 (92.13%)</td>
</tr>
<tr>
<td>Secondary and Tertiary searches</td>
<td>32</td>
<td>5 (15.63%)</td>
<td>27 (84.38%)</td>
</tr>
<tr>
<td>Total</td>
<td>568</td>
<td>52 (9.15%)</td>
<td>516 (90.85%)</td>
</tr>
</tbody>
</table>

3.2.2. Study selection. After retrieving all the unduplicated population of scientific papers relevant to the research questions, we selected the publications relevant to the review objective according to the inclusion and exclusion criteria defined in Section 3.1.3. On evaluating the title and abstract, 168 studies were rejected. After a quick read-through the bodies of the papers, 150 studies were rejected. Therefore, a total of 516 studies were rejected (see Table III). After this process, a total of 52 relevant studies (see Table III) were entered to a spreadsheet, along with their bibTex source, and electronic versions of publications were also stored in a filesystem directory for easy access during the systematic review.

3.2.3. Study quality assessment, data extraction and data synthesis. Each selected study was assessed on the basis of the quality criteria defined in Section 3.1.4; 75% of studies passed the defined quality criteria (39 scientific papers). The data extraction consisted of coping verbatim aims, findings, and conclusions reported by the studies into a commercial tool for organizing and tracking information. Finally, results were synthesized in tabular forms that show the APLE roadmap. They allow us to compare across studies and to identify current challenges and gaps (see Section 4).

4. THIRD PHASE: REPORTING THE REVIEW

The systematic review retrieved 39 scientific papers which have been identified from P1 to P39 in chronological order of publication (see Table IV). They have been obtained from different
publication channels Table V shows. The research questions defined in Section 3.1.1 have been addressed through the 39 selected papers in this systematic review.

Table IV. List of the correspondences between the selected papers and the references in chronological order.

<table>
<thead>
<tr>
<th>Id</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>[18]</td>
</tr>
<tr>
<td>P2</td>
<td>[4]</td>
</tr>
<tr>
<td>P3</td>
<td>[25]</td>
</tr>
<tr>
<td>P4</td>
<td>[7]</td>
</tr>
<tr>
<td>P5</td>
<td>[32]</td>
</tr>
<tr>
<td>P6</td>
<td>[36]</td>
</tr>
<tr>
<td>P7</td>
<td>[40]</td>
</tr>
<tr>
<td>P8</td>
<td>[44]</td>
</tr>
<tr>
<td>P9</td>
<td>[47]</td>
</tr>
<tr>
<td>P10</td>
<td>[51]</td>
</tr>
</tbody>
</table>

Table V. Distribution of studies according to the publication channel.

<table>
<thead>
<tr>
<th>Publication channel</th>
<th>Type</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>APLE Workshop</td>
<td>Workshop</td>
<td>7</td>
</tr>
<tr>
<td>SPLC Conference</td>
<td>Conference</td>
<td>6</td>
</tr>
<tr>
<td>Journal of System and Software</td>
<td>Journal</td>
<td>4</td>
</tr>
<tr>
<td>XP</td>
<td>Conference</td>
<td>4</td>
</tr>
<tr>
<td>CEE-SET Conference</td>
<td>Conference</td>
<td>3</td>
</tr>
<tr>
<td>Agile Conference</td>
<td>Conference</td>
<td>2</td>
</tr>
<tr>
<td>Journal of Object Technology</td>
<td>Journal</td>
<td>2</td>
</tr>
<tr>
<td>PROFES Conference</td>
<td>Conference</td>
<td>1</td>
</tr>
<tr>
<td>VaMoS Workshop</td>
<td>Workshop</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Total</strong> 39</td>
</tr>
</tbody>
</table>

4.1. **RQ1**: What are the reasons for combining SPLE and ASD? When is it advantageous to put APLE into practice?

Several studies have conclude that there are sufficient reasons to move towards a combination of SPLE and ASD. The main three aims that lead to combination of SPLE and ASD are:

1. To cut down the long-term investment during the DE phase. Känsälä (P2) and Taborda (P3) identify the challenges to be faced by SPL in volatile, dynamic market conditions. DE requires to completely defin and plan the commonality and variability for all products of the SPL. It means an upfront long-term investment in designing a set of core-assets/platform (P18) [3]. The upfront long-term design results in a SPL platform that is flexible in the sense that it is capable of accommodating variation for those anticipated changes, i.e. the identified variabilities across the products of the SPL. Companies can leverage this investment when new products are developed through systematic reuse of core-assets by deriving the SPL variability points with the selected variants. Although it has been considered successful [2], this strict “domain-then-application” model is resource intensive and risky (P28). Since core-assets are planned with a long term perspective, there is a risk of developing core-assets that will become obsolete and never used in product derivation.

2. To deal with volatile business situations. When market stability decreases, the risk of developing core-assets that will become obsolete and never used in product derivation increases. Then, the upfront effort for planning and designing SPLs might be wasted in
changing market situations (P3). Känsälä (P2) defines a software process knowledge-base for Nokia, called *Business Framework for SW Process* to map software processes to the size and complexity of software and business volatility. This framework identifies a *sweet zone*, where traditional approaches tackle the increasing size and complexity of software through SPLE, and a *difficult zone*, where large/complex software projects have to deal with volatile business situations. According to Känsälä, business volatility requires alternatives to supplement SPLE, and ASD is proposed as a promising alternative.

3. To deal with situations in which there is no knowledge enough about DE. DE and the resulting built-in flexibility have proven to be a crucial success factor in contexts in which anticipated changes can be predicted with certain accuracy. If SPL developers do not have knowledge enough to perform the DE. ASD may facilitate the elicitation of further knowledge (P8). ASD expects limited predictability targeting on smaller-scale development efforts, assuming that future changing scenarios cannot be anticipated (P18). As a result, when anticipated changes cannot be predicted and the software product life-cycle is not known, it would be advantageous to use an incremental approach such as APLE (P5).

Therefore, the long-term investment that is carried out during the DE phase of a typical SPL development, could be replaced by an agile approach when changes cannot be anticipated. Changes cannot be anticipated due to whether volatile business situations or the knowledge on DE is not enough. Since efforts are focused on limited predictability, risks are reduced. According to (P8), trade-offs between SPLE and ASD provide the opportunity to apply the APLE approach to a wider variety of projects than those served by ASD or SPL methods. This is the case for critical systems, large-scale systems, or those with distributed development teams. Most of the authors agree that SPLE and ASD are complementary to deal with the issues of large-scale product families in volatile markets (P2-P5, P8, P9, P11, P14, P18, P20, P22, P24-39).

4.2. **RQ2: How do the principles of SPLE and ASD match with each other?**

Tian et al. (P8), Hanssen et al. (P14, P18, P38), and Mohan et al. (P39) compare SPLE and ASD with the aim of determining the feasibility and conflicts of bringing together both approaches. Although both SPLE and ASD pursue common goals such as improving customer satisfaction and flexibility, and reducing time-to-market, SPLE and ASD apply different strategies to achieve these goals (P4-P10, P18, P39). On the one hand, SPLE stresses the importance of predicting changes at the beginning of the process, and the need of defining a PL Architecture to support customization; in this sense it is capable of accommodating predicted changes to potential members of the product line. On the other hand, ASD emphasizes value delivery to the customer and welcomes changes by means of incremental development and close iterations with customers. However, ASD methods (e.g. XP and Scrum) advocate for minimal investment in an upfront architecture when knowledge is not readily available, and encourage the continuous improvement and refactoring of the architecture to achieve the business goals. In fact, agile methods have a reputation for paying very little attention to software architecture (P18) and some agile practitioners even advocate against investing effort in architecture specification as it is perceived as wasted effort [53].

McGregor (P31, P32) presents a theoretical attempt to reconstruct a hybrid method from SPLE and ASD principles. He concluded that although the integration of both paradigms is difficult, SPLE and ASD can be tailored under the condition that both should retain their basic principles. Following this same idea, Hanssen et al. (P18) assert that, although at first sight SPLE and ASD may seem contradictory, they actually complement each other. An overall mapping between SPLE and ASD principles is carried out by Hanssen et al. (P18) to identify similarities, but also differences between both the approaches. Table VI summarizes this mapping, but the complete mapping with the 12 agile principles can be found in (P8). If we focus on principles, it seems feasible to tailor SPLE with ASD (P8) to obtain an approach that (i) analyzes the most significant commonalities in a domain, rather than an exhaustive set; and (ii) meets changing customer requirements, rather than just simply customizing core-assets.
Table VI. Matching ASD and SPLE. Adapted from [P18].

<table>
<thead>
<tr>
<th>Agile Principles</th>
<th>SPLE correspondence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deliver working software frequently</td>
<td>Product delivery might be slowed down compared to ASD due to overhead in maintaining integrity with the SPL platform</td>
</tr>
<tr>
<td>Welcome changing requirements</td>
<td>Changes planned in variability models are cheaply and quickly implemented (configurability)</td>
</tr>
<tr>
<td>Continuous attention to technical excellence</td>
<td>SPL platforms encode technical excellence and design to support rapid product derivation</td>
</tr>
<tr>
<td>The best architectures, requirements, and design emerge from self-organizing teams</td>
<td>Investments in requirements, architectures, and designs can be reused across multiple products</td>
</tr>
<tr>
<td>Continuous delivery of valuable software</td>
<td>Dependent upon SPLE approach</td>
</tr>
</tbody>
</table>

4.3. **RQ3**: How are SPLE and ASD positioned with respect to business strategic objectives?

This question is especially significant, since it is close to the organization top management, which will provide the basic operations guidelines. A very high level, both SPLE and ASD, by definition, are aligned with business strategy: SPLE scoping is obtained from this business strategy; ASD claims that value delivery to the customer is an utmost objective, and customer participation is a usual practice. Strategic objectives are dealt with by (P18, P39). While both combine SPLE and ASD, they follow a different approach with respect to strategic objectives. In (P18), SPLE is used at a strategic level while ASD is used at a tactical level, i.e. project management. (P39) highlights the importance of top management to (i) manage the relationships among technological, organizational, and strategic objectives, (ii) continuously assess the alignment of these objectives, and recognizing the evolutionary nature of these relationships; and (iii) cope with a considerable flexibility in development processes. These two approaches (P18) and (P39), while radically different, provide a good basis for their justification. Hanssen et al. (P18) work with the hypothesis that ASD is useful for the tactical level, which is to assume that the scope of ASD planning is short term; Mohan et al. (P39) recommend having ASD at strategic level, that is mid/long term planning, to foster flexibility. Examples of deploying the tactical level can be found in (P3, P6, P16). This different interpretation and scoping on how to take SPLE and ASD into practice, finds an explanation and guidance in (P35). Codenie et al. (P35) explain how it can make sense considering: the kind of application, all kind of customer needs, strategic objectives, and the flexibility degree of the approach. That is, organizations should analyze the consequences of deciding a higher or lower level of mass customization and/or flexibility in terms of investment, and business, organizational and technological factors.

From these studies we can conclude that the current literature has addressed the theoretical mapping between the principles of both SPLE and ASD, and how these principles can live together. All authors agree that trade-offs, specific for each project, are necessary to successfully support APLE.

4.4. **RQ4**: Which current approaches combine SPLE and ASD satisfying Application Engineering activities? **RQ5**: What are the challenges and gaps in current approaches that combine SPLE and ASD during Application Engineering activities?

The research work of Taborda (P3) proposes the implementation of a release matrix to support Release Planning and tracking of SPLs in ASD. This release matrix allows us to plan and track the evolution of the PL Architecture over time in an incremental release strategy, addressing multiple products, components and their inter-dependencies. However, it lacks formalization, and deals with release planning documentation rather than how to deal with product release.

Together with Release Planning, Configuration Management (CM) is a key activity to ensure successful SPLs. The management, maintenance, and tracking of multiple artifacts derived from a SPL requires a substantial and complex effort. Therefore, more powerful and robust CM tools than
those for a single-system development are required. Kurman (P6) addresses this problem arguing that: (i) automatic product configuration is a key issue for APLE, and (ii) automatic building and deployment have to be 100% reproducible at any time.

Carbon et al. (P5) strongly believe that SPLE and ASD must be combined, arguing that ‘a combined SPLE approach would thus build-in flexibility for the most probable anticipated changes (proactive), while being agile enough to quickly incorporate changes that were not anticipated (reactive) and for no flexibility was built in’. The authors describe how Agile methods are used in a product derivation process called PuLSE-I [54], where planning games and incremental design are normal practices. Software products are developed in such a way that a first version includes the core functionality, and further iterations add new functionality. It satisfies the agile principle of *early and continuous delivery of working software*. The authors also highlight an experiment to study flexible design up-front vs. incremental design from a return on investment perspective. However, the experiment results are not described in detail. In a later work, Carbon et al. (P22) stress the importance of feedback from application to domain engineering in order to minimize the degeneration of the product line infrastructure, and thus maximize its viability. The feedback mechanism proposed by Carbon et al. is the planning game, where application engineers take over the role of customers and domain engineers take over the role of developers. The former specify suggestions to the latter about how improving the reusability of core-assets for specific product applications.

O’leary et al. (P24-P27) provide a much more comprehensive framework to cover all activities for Product Derivation (PD) than the previous authors. They define an Agile Framework for Product Derivation (AFPD) which assists organizations in using a structured approach for PD activities and provides a means of integrating Agile practices in the PD process. AFPD considers the adoption of early and continuous delivery strategy, automation of product derivation, product derivation iterations, and agile testing techniques. The O’leary’s work on SPLE area has provided a set of key activities, that any PD approach should consider, and a set of issues to be addressed [55] in a short-medium term. These issues are related to provide mechanisms for supporting effort estimation, traceability, synchronisation between platform and product teams (DE and AE teams, respectively), and how representing product line variability from different points of view, for instance, *how to represent variability in the language of the product derivation stakeholders*. These issues include a number of challenges not yet completely solved for SPL.

4.5. **RQ6**: Which current approaches combine SPLE and ASD satisfying Domain Engineering activities?  
**RQ7**: What are the challenges and gaps in current approaches that combine SPLE and ASD during Domain Engineering activities?

According to the panel “Agile and Product Line Engineering”, APLE’08 (P20), it was not clear how agile methods could be applied to DE. Nevertheless, some attempts have been made to address its applicability.

The work of Noor (P7), Feng (P13), and Trinidad (P17) tackle very specific DE activities such as product-line scoping, requirements engineering or feature modeling. Noor et al. (P7, P15, P16) provide an approach to collaboratively support the product line scoping activity, which includes agile principles such as stakeholder involvement, rapid feedback, or value-based prioritization. Feng et al. (P13) provide some initial results on how selecting a requirements engineering (RE) process, which provides the necessary degree of agility and the support for specifying a product line application. They gather the knowledge that influences the RE process selection with a tailored level of agility. Trinidad et al. (P17) present an automated error analysis method to detect errors (e.g. dead features) in feature models. Since feature models are usually strongly affected by changes in requirements, and one of the objectives of agile methods is a rapid response to requirement changes, (P17) makes more agile feature modeling.

The work of Paige (P11) and Kakarontzas (P23) tackle the design of PL Architectures. Paige et al. (P11) propose building software product lines using the agile method Feature Driven Development (FDD), in such a way that SPL development process can benefit from agile development techniques to identify SPL features, configurations, and variability points. Hence, the main challenge is how...
to integrate PL Architecture design into FDD. To deal with this challenge, Paige et al. propose that architecture should be incrementally designed (iterative architecture⁴), and SPL variations should be generated as a result of the agile refactoring practice. Even though this proposal of SPL and FDD integration shows potential and is promising, no concrete data, process description, or cases were presented. Moreover, considerations about incremental design of PL Architecture were not described either. Meanwhile, Kakarontzas et al. (P23) present an approach based on Test Driven Development (TDD) and elastic components to specify product line variability. Elastic components address the configuration and evolution of components by means of adding/deleting/modifying variants that hook from the root component. TDD ensures that the functionality and quality of the product is not degraded during evolution. This work is a first step to show how PL Architectures can be tailored to be more Agile. However, since their variants are context-dependent, additional mechanisms are needed to make variants independent of the root component. Context-independent variants may (i) facilitate the incremental and iterative evolution of PL Architectures in ASD by reducing the number of changes; and (ii) make more flexible the reuse of variants among the products of a SPL.

Finally, Raatikainen et al. (P21) introduce a tool based approach to monitor, plan and measure the development process of a SPL. This tool integrates Kumbang (a feature modeling tool), and Agilefant (an agile backlog tool). Even though the combination looks promising, no concrete data or cases were presented and the agility of the method was not described.

4.6. RQ8: Which current approaches combine SPLE and ASD satisfying both Domain Engineering and Application Engineering activities through Agile Principles?

The integration of SPLE and ASD principles and practices, satisfying both DE and AE, is really challenging, and it is only addressed by the work of Ghanam et al. (P28-P30, P37). While all previous works target SPL-based organizations that want to make their product development more agile, Ghanam et al. (P28-P30, P37) target agile organizations that wish to establish a SPL. The authors highlight the importance of mining systems, bearing in mind reuse, and the formalization of commonality and variability through acceptance tests. Hence, they introduce an iterative model for APLE and the use of TDD to support this process. Specifically, they propose a bottom-up application-driven approach that relies on automated acceptance tests to derive core-assets from existing code. Their approach uses executable acceptance tests to support variability management and traceability: an acceptance test has a generic fixed component and a variable component. To bridge the gap between SPLs and the XP agile method, executable acceptance tests are used to describe the system and act as anchor points for traceability relations. Through a bottom-up approach, the SPL is iteratively built from existing product instances, the SPL platform evolves progressively in an iterative approach, and variability is handled on-demand, i.e. reactively. (P37) presents a tool for assisting the refactoring of code when a developer introduces a variation. The code-based tool requests to developers the method that is causing the variation and creates an abstract factory for this method, the corresponding concrete classes, and the required test classes. This approach provides a novel contribution based on TDD to model variability dealing with DE and AE in a context where the XP method is used. Despite the fact that this contribution is a significant advance in the area, more work is still necessary to support coarse-grained variability, i.e. not always it is possible to trace a variation with a class or a method. A variation could be traced to a set of methods, classes, attributes, or even components.

4.7. RQ9: Are there successful industrial experiences putting APLE into practice?

Kane (P1) describes his experience incrementally introducing agile development techniques into a bioinformatic institute. This experience report identifies several challenges, including how better

⁴An iterative architecture is one that develops with the system, and includes only features that are necessary for the current iteration or delivery. The architecture is a working vision for developers, which encapsulates important features of the existing design [56]
managing a SPL with agile techniques and, specifically how balancing the priorities between the products of a SPL. Another industrial experience, by Hanssen et al. (P18), describes a successful case study of integrating SPL and ASD practices. They use SPL practices as a strategic platform for developing products with the agile EVO method [57]. Kircher et al. (P12) identify a collection of successful best practices when they introduce a SPL approach in Siemens business groups. They assert that SPLs cannot be considered in isolation as the usage of agile processes may facilitate fast feedback cycles between requirements engineering, development, and field trial, in innovative business. A key issue that is not described, is how a SPL approach integrates the best practices of Agile methodologies.

Another empirical study was developed by Ali Babar et al. (P36) who describe a successful industrial case study, and analyze the organizational processes and practices that were used to integrate SPLs and ASD. The authors describe a development process that consists of three sub-processes: product line platform, exploration before agile product development, and agile product development. It was found that architecture and architectural communication support these three processes and, reciprocally, these processes may update product architectures by means of refactoring. Although this approach is valid and compelling, it focuses on a reuse-centric AE process, and does not discuss the role of agile methods in DE. As a result, it is closer to the concept of agile software development using a product line platform than the concept of APLE.

5. KEY FINDINGS: IMPLICATIONS FOR PRACTITIONERS AND RESEARCHERS

From this systematic literature review two main ideas can be highlighted.

- First, practitioners can conclude that there are sufficient reasons to move towards a combination of SPL and ASD.
- And second, researchers can conclude that there are still some important challenges in the area, and therefore, more research work is required to completely put into practice APLE.

With regard to the first idea, the literature review reports that APLE would be applicable to business situations in which the convenience of going towards a product line has been identified, but at the same time the market situation is not enough stable for different reasons, including technological factors. Specifically, the review has identified four main advantages of putting APLE into practice, and when it is advantageous:

1. If SPL developers do not have enough knowledge to completely perform the DE, ASD may facilitate the elicitation of further knowledge (P8).
2. Trade-offs between SPL and ASD provide the opportunity to apply the APLE approach to a wider variety of projects than those served by only applying ASD or SPL methods (P8).
3. When anticipated changes cannot be predicted and the product life-cycle is not known, it would be advantageous to use an incremental approach such as APLE (P5).
4. Agile processes may facilitate fast feedback cycles between requirements engineering, development, and field trial in innovative business (P12).

With regard to the second idea, the literature review has reported four main findings:

1. The applicability of agile methods to DE requires more effort than the application of AE. This is due to the fact that is difficult to reduce the upfront design with the aim of getting closer to agile principles and values, while achieving the typical goals of DE such as reuse (P33). Nevertheless, the applicability of agile methods to AE does seem feasible. The works by Carbon et al. (P5, P22) and O’Leary et al. (P24-P27) provide a complete support for product derivation.
2. Synchronization between platform and product teams (DE and AE teams, respectively) is vital in APLE, as DE and AE should not be separated. The platform should be synchronized with the application needs to avoid that the platform will become obsolete (P26). It is still a challenge for APLE engineers (P33).
3. Business objectives should be used to identify the right level of flexibility. This level should be useful to determine the combination of SPLE and ASD: either SPLE and ASD at strategic level, or SPLE at strategic level and ASD at tactical (P8, P18, P39).

Since this systematic literature review pursues to be useful for both practitioners and researchers, the analysis of the most relevant findings have been classified into two points of view: (i) How can APLE be put into practice?, i.e. what problems have been solved in APLE deployment and which approaches/practices can be considered for APLE adoption; and (ii) Which are the open research challenges?, i.e. what is still missing and what challenges/barriers researchers should cope with? The analysis from these points of view, is detailed in the following subsections.

5.1. How can APLE be put into practice?

Table VII shows a global view of the contributions and approaches (exactly 15 approaches) that practitioners interested in APLE can put into practice in their developments. This table highlights practices and experiences (30 approx.) which completely or partially solve the most recurrent challenges in APLE. Table VII also classifies these practices and experiences based on: the process/es (DE or AE) they address, which specific activity they deal with, and which research question of this systematic literature review they answer. For each challenge that has been identified in this systematic literature review, we have categorized these contributions and approaches as follows:

1. Whereas SPLE stresses the importance of the upfront investment, planning and design, ASD emphasizes delivering value to the customer rapidly and frequently, advocates welcoming changes, and has the reputation for paying little attention the design. As a result, their combination is a challenge. Five contributions/approaches address this challenge. Specifically, the work of Hanssen et al. (P14, P18, P38) and Ghanam et al. (P28-P30, P37) provide industrial case studies (P18) and tool support (P37) on applying XP and EVO agile methods in SPL development.

2. The welcome to changes that ASD suggests, requires mechanisms to support it. Three contributions/approaches address this challenge by means of mechanisms to deal with the change in feature models (P10, P17), product line scoping (P7, P15, P16), and requirements engineering processes (P13). These contributions provide successful industrial case studies and tool support. However, how changes are managed in successive activities such as architecture, design, product derivation, etc. is not dealt with.

3. As SPLE is tailored in favor of an agile incremental development in short iterations, the PL Architecture must also support its incremental design. Two contributions/approaches address this challenge (P11, P23), but they do not report case studies to illustrate their applicability.

4. As SPLE is tailored in favor of an agile incremental development in short iterations, the product derivation process must also support the frequent delivery of value to the customer. Five contributions/approaches address this challenge (P3, P5, P6, P24-P27, P36). These contributions provide successful industrial case studies and tool support for the entire product derivation process and specific activities such as release planning or product configuration.
Table VII. Summary for contributions/approaches working on a challenge.

<table>
<thead>
<tr>
<th>Challenges</th>
<th>Contributions &amp; Approaches</th>
<th>Practices</th>
<th>Experiences</th>
<th>DE/ AE</th>
<th>Activities</th>
<th>Research Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>A business framework maps software process with software size and complexity and business volatility (P2).</td>
<td>–</td>
<td>Nokia experience</td>
<td>–</td>
<td>–</td>
<td>RQ1</td>
<td></td>
</tr>
<tr>
<td>A lighter-weight approach by tailoring a SPL approach with Agile practices (P8).</td>
<td>Analysis of the most significant commonalities in a domain, rather than an exhaustive set. Welcome to change customer requirements, rather than just simply customization.</td>
<td>–</td>
<td>Industrial case study</td>
<td>–</td>
<td>All</td>
<td>RQ2</td>
</tr>
<tr>
<td>Combination of the upfront investment, effort, planning and design of SPL and the incremental and “welcome to changes” development of ASD (P18, P19, P32).</td>
<td>Fusion process of SPL and ASD using the agile method EVO (P14, P18, P38).</td>
<td>SPL addresses strategic long-term objectives and ASD addresses tactical short-term objectives.</td>
<td>Tool + case studies.</td>
<td>DE/ AE</td>
<td>All</td>
<td>RQ8</td>
</tr>
<tr>
<td></td>
<td>An APLE approach based on an iterative model and the agile method TDD (P28, P30, P37).</td>
<td>Bottom-up application-driven approach that relies on automated acceptance tests (ATs) to derive core-assets from existing code. Formulation of commonality and variability through ATs. ATs act as anchor points for traceability relations. Agile refactoring to support variations.</td>
<td>Tool + case studies.</td>
<td>DE/ AE</td>
<td>All</td>
<td>RQ8</td>
</tr>
<tr>
<td></td>
<td>Approach to reconcile the Agile and CMMI Contexts in Product Line Development (P9).</td>
<td>SPL, CMMI and Agile</td>
<td>–</td>
<td>DE/ AE</td>
<td>Orthogonal</td>
<td>RQ9</td>
</tr>
<tr>
<td>Rapid response to changes in requirements</td>
<td>An automated error analysis to detect errors in feature models due to changes in requirements (P10, P17).</td>
<td>Release planning and tracking by means of a Release Matrix.</td>
<td>Tool + industrial case study</td>
<td>DE</td>
<td>Feature Modeling</td>
<td>RQ6</td>
</tr>
<tr>
<td></td>
<td>Collaborative support for the product line scoping activity (P7, P15, P16).</td>
<td>Stakeholder involvement, rapid feedback, or value-based prioritization.</td>
<td>Tool + industrial case study</td>
<td>DE</td>
<td>Product Line Scoping</td>
<td>RQ6</td>
</tr>
<tr>
<td></td>
<td>A framework to select a RE process with the necessary degree of agility (P13).</td>
<td>A survey to collect expertise in APLE RE</td>
<td>–</td>
<td>DE</td>
<td>Requirements Engineering</td>
<td>RQ6</td>
</tr>
<tr>
<td>Incremental PL Architecture</td>
<td>An APLE approach based on an Iterative model and the agile method FDD (P11).</td>
<td>PL Architecture is realized incrementally, and SPL variations are generated as a result of the agile refactoring practice.</td>
<td>Case study.</td>
<td>DE</td>
<td>PLA Design</td>
<td>RQ6</td>
</tr>
<tr>
<td></td>
<td>Approach based on TDD and elastic components to specify variability in PL Architectures (P23).</td>
<td>Elastic components address the configurability and evolution of components. TDD ensures that the functionality and quality of the product is not undermined during the evolution.</td>
<td>–</td>
<td>DE</td>
<td>PLA Design</td>
<td>RQ6</td>
</tr>
<tr>
<td>Agile Product Configuration and Derivation</td>
<td>Plan and track the evolution of a PL Architecture over time in an incremental release strategy (P3).</td>
<td>Release planning and tracking by means of a Release Matrix.</td>
<td>Trials of the Release Matrix in a large software development organization.</td>
<td>AE</td>
<td>Release Planning</td>
<td>RQ4</td>
</tr>
<tr>
<td></td>
<td>Automatic product configuration (P4).</td>
<td>SPL approach acts as an overall development strategy and is combined with some agile software development techniques: daily user deployment test, welcome to changes, etc.</td>
<td>Industrial case study.</td>
<td>AE</td>
<td>Release Planning and Product Configuration</td>
<td>RQ4</td>
</tr>
<tr>
<td></td>
<td>Development process made of three subprocesses: product line platform, exploration before agile product development, and agile product development (P16).</td>
<td>Architecture and architectural communication support these three processes and, reciprocally, these processes may update product architectures by means of refactoring.</td>
<td>Industrial case study.</td>
<td>DE/ AE</td>
<td>Agile is mainly involved in product derivation.</td>
<td>RQ9</td>
</tr>
<tr>
<td></td>
<td>Agile methods as a product derivation process called PDLSEF (P5).</td>
<td>Incremental design (early and continuous delivery of working software) Feedback from AE to DE to minimize the degeneration of the product line infrastructure, and thus maximize its viability. The feedback mechanism is the agile practice ‘planning game’.</td>
<td>Experiment: Flexible Design Up-Front vs. Incremental Design.</td>
<td>AE</td>
<td>All</td>
<td>RQ4</td>
</tr>
<tr>
<td></td>
<td>Agile Framework for Product Derivation (AFPD) (P24-27).</td>
<td>Adaption of early and continuous delivery strategy, automation of product derivation, product derivation iterations, and agile testing techniques.</td>
<td>Industrial case study.</td>
<td>AE</td>
<td>All</td>
<td>RQ4</td>
</tr>
</tbody>
</table>

4 challenges 15 contributions/approaches 30 practices approx. 7 industrial CS, 3 tools, 3 CS/trials, and 1 survey 10 DE 8 AE
5.2. Which are the open research challenges?

Tables VIII and IX set out the activities and practices for AE and DE, respectively. These tables reveal that there exist gaps and more research is required. These tables only consider the subset of papers that is relevant to the activities and practices being analyzed. Each row represent the work of an author that is relevant to a specific activity/practice, and the last row show the total number of papers that address each activity/practice. The ✓ mark means that authors explicitly propose a solution to support an activity; the x mark means that authors explicitly identify a challenge or gap in that activity; finally the - mark means there is no evidence.

In view of Table VIII, the applicability of agile methods to AE seems to be really feasible. Twelve papers address AE, specifically ten of them propose solutions related to release planning, 11 of them are related to product configuration, 10 are related to product development, 10 are related to test, integration and deployment, 6 related to traceability, and 10 of them are related to maintenance and evolution. Only one gap has been identified by the papers included in this systematic review. From the 6 papers that provide solutions for traceability, 4 papers report the problem to define traceability among the different SPL artifacts and the synchronization among product and platform teams.

In view of Table IX, the applicability of agile methods to DE presents lack of evidence for most of the activities/practices. Nevertheless, some solutions are provided: three papers tackle SPL scoping, one paper addresses requirements engineering, seven papers tackle feature model, and four papers address traceability. However two activities deserve special attention: architecture and traceability. Tian et al. (P8) explicitly recognized potential challenges and risks associated with APLE: (i) traceability management and maintenance of components might be difficult in agile approaches without explicit knowledge; and (ii) if PL Architectures are tailored to be more Agile, there is a danger that a valuable architecture supporting other products of the family may be damaged. As a result, APLE architecture and APLE traceability are still open research challenges.

With regard to the traceability challenge, work in progress (P28-30) has tackled traceability management. This work is inclined toward TDD approaches to take advantage of iteratively track and test software. However, it is still necessary additional effort to be able to support traces between features and core-assets to easily implement maintenance tasks in a systematic and (semi)-automatic way.

With regard to the architecture challenge, most approaches converge towards iteratively and progressively build the SPL platform (P8, P11, P18, P29, P30), and specifically, the PL Architecture (P11, P23). In practical terms, the PL Architecture should have mechanisms to support evolution, i.e. to flexibly be adapted. But, the agile design of PL Architectures has been pointed out as a key challenge to overcome, since none of the mentioned authors have completely solved it. Dealing with this challenge is essential with the aim of APLE can be widely adopted by software industry (P8, P18, P29, P36). The key question is: how to take care of the long-term planning and upfront architecture required by SPL while still being able to deliver value to the customer in time? (P36); or from the opposite point of view: how architecture can be tailored to be more agile without losing the SPL reusability and flexibility? It should not be forgotten that software architectures are a key factor in the success of SPLs, while Agile considers that architecture will emerge gradually in an iterative and incremental way. Thereby, Agile will pay attention to design, and much more attention to early delivery of valuable software (P18), putting in a second place customization (P8). Following (P33), it may be useful to have a mechanism to specify qualities for a good PL Architecture that will allow us to perform a trade-off between upfront long-term design of SPLs and ASD short-term design.

5.3. Final considerations

To keep this systematic literature review up to date of publication, we have made an effort during the review process to include the last papers and advances on APLE. Although these papers have not been included in the systematic analysis, it is important to mention some new approaches and techniques that contribute to put APLE into practice. In [6], the authors propose a framework to incrementally and reactively construct variability profiles for existing and new systems. The framework leverages common agile practices such as iterative software development, refactoring,
Table VIII. Summary for Application Engineering.

<table>
<thead>
<tr>
<th>References</th>
<th>Release Planning</th>
<th>Product Configuration</th>
<th>Product Development</th>
<th>Test, integration, deployment</th>
<th>Traceability and evolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P3)</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>(P5,P22)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>(P6)</td>
<td>✓</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>(P24-27)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>(P28-30,P37)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

| 12          | 10               | 11                    | 10                  | 10                          | 6                          |

Table IX. Summary for Domain Engineering.

<table>
<thead>
<tr>
<th>References</th>
<th>Scoping Engineering</th>
<th>Requirements Engineering</th>
<th>Feature Model</th>
<th>Architecture Core Assets Development</th>
<th>Traceability Evolution and Maintenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P8)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>(P7,P15,P16)</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(P10,P17)</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(P14,P18,P38)</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(P11)</td>
<td>-</td>
<td>✓</td>
<td>x</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(P13)</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(P23)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(P24-27)</td>
<td>-</td>
<td>✓</td>
<td>x</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(P28-30,P37)</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

| 20          | 3                 | 1                       | 7               | 0                                     | 4                          |

continuous integration and testing to introduce variability into systems only when it is needed. In [58], the authors attempt to solve the problem of designing software architectures in Agile based on the Plastic Partial Component (PPC) concept. PPCs are highly malleable components that can be partially described, what increases the flexibility of architecture design. In fact, the notion of PPC was originally defined for SPL to support the definition of internal variation of architectural components [59]. Hanssen et al. [60] report how agility and entropy∥ are negatively related. Although their contribution is not specific for SPL, otherwise for product evolution in general, a case study shows (i) how a successful software product line organization has adopted the agile development method Evo [57]; and (ii) how to manage entropy while the process maintains agility. Finally, in [61], the authors propose a traceability mechanism to ensure consistency between feature models and code artifacts during evolution processes where Agile Software Development is practiced.

6. CONCLUSIONS

This paper presents the results of a systematic literature review of practices and experiences on APLE. Although some authors reveal some disagreements about the combination of SPL and ASD, practically all of our findings agree that SPL and ASD can work together since they pursue the same high level goals but using different methods to achieve them. It has been acknowledged that the main advantage of APLE is that SPL may be applied to volatile markets in which changes cannot be predicted easily, supported by ASD to develop large-scale product families (SPLs). A significant issue is to identify the right level of flexibility according the business objectives. This will be the basis to decide if the approach is SPL and ASD at strategic level, or SPL at strategic and ASD at tactical.

∥This concept is defined in [60] as how the maintainability of a system may degrade over time due to continuous change.
Therefore, practitioners can conclude that there are sufficient reasons to move towards a combination of SPLE and ASD. However, there are still some important challenges in the area, and therefore, more research work is required in order to completely put into practice APLE.

ACKNOWLEDGMENT

This work has been partially sponsored by the Spanish MEC (DSDM TIN2008-00889-E), MICINN (INNOSEP TIN2009-13849), MITYC (FLEXI FIT-340005-2007-37, ITEA2 6022) and UPM (Technical University of Madrid) under their Researcher Training program.

REFERENCES


