Methodologies for Agile Product Line Engineering: A Survey and Evaluation

CONFERENCE PAPER · SEPTEMBER 2014

DOI: 10.3233/978-1-61499-434-3-545

2 AUTHORS, INCLUDING:

Farima Farmahini Farahani

Sharif University of Technology

1 PUBLICATION **0** CITATIONS

Methodologies for Agile Product Line Engineering: A Survey and Evaluation

Farima Farmahini FARAHANI[1] and Raman RAMSIN
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

Abstract. Agile Product Line Engineering (APLE) is a relatively new approach which has emerged as the result of combining two successful approaches: Software Product Line Engineering and Agile Software Development. The goal of this combined approach is to cover the weaknesses of each of the two approaches while maximizing the advantages of both. Several methodologies exist which provide a practical process for applying APLE in organizations. In this paper, these APLE methodologies will be evaluated using a criteria-based approach. Results of this evaluation show each methodology's strengths and weaknesses, and can be helpful in selecting, comparing, and modifying APLE methodologies. The evaluation framework and the results can also be used for developing bespoke APLE methodologies, tailored to fit the specific needs of organizations and individual projects.

Keywords. Software development methodology, product line engineering, agile software development, criteria-based evaluation

Introduction

The software industry has always been seeking for ways to accelerate the delivery of high quality products while reducing development costs. To achieve these goals, several methods and approaches have been proposed by researchers and practitioners. Among the most successful approaches are "Agile Software Development" and "Product Line Engineering (PLE)". Both of these approaches fulfill the mentioned goals, albeit through different strategies, and this has resulted in their popularity among software developers. The successful results of applying these approaches have motivated researchers to find ways for merging them; the approach which has emerged as the result of this merger is called "Agile Product Line Engineering (APLE)".

The ultimate goal in APLE is to maximize the benefits of each of the individual approaches and to fulfill their common goals. These common goals are: Managing changes in requirements, promoting product quality, decreasing development costs, and reducing time to market. Another significant advantage in combining the agile and PLE approaches is synergy: Each approach has the capacity to address the weaknesses of the other. Although there are many advantages in combining the two approaches, certain difficulties also exist, mainly due to the inherent differences of the two approaches. These differences include: Different strategies for handling changing requirements,

[1] Corresponding Author: Farima Farmahini Farahani, Department of Computer Engineering, Sharif University of Technology, Azadi Ave., Tehran, Iran; E-mail: farimafarahani@ce.sharif.edu

difference in the degree of focus on documentation, disagreement as to the level of user involvement required, and different development roles involved [1].

Several methods have so far been proposed to address these challenges and to effectively merge the two approaches. From among these methods, only a relatively small number have proposed a *process* for this combined approach, and can hence be referred to as APLE *methodologies*. As frameworks for organizing software development activities and practices, software development methodologies consist of two integral parts: A modeling language and a process [2]. The modeling language part provides the syntax and semantics used for expressing the products, whereas the process prescribes the flow of activities that should be performed and explains how the products should be produced, enhanced and exchanged along this flow. The agility feature of APLE methods has deemphasized the role of modeling, and hence the modeling language, in such methods; therefore, the main distinctive feature which distinguishes an APLE methodology from the simple methods used in this context is that a methodology incorporates a distinct process.

This paper focuses on studying and evaluating the APLE methodologies which have been introduced so far. Several APLE methods have already been surveyed in [3], but our intention has been to survey and analyze APLE methodologies in a more precise and systematic manner through using criteria-based evaluation. As the first step of this research, current APLE methodologies were identified and studied; the first version of evaluation criteria was then developed based on the characteristics elicited from the studied methodologies. The main stage of the evaluation process was then carried out by iterative evaluation of the methodologies based on the criterion set. The results of the evaluation performed in each iteration provide a deeper insight into the features of the methodologies, and can thus draw attention to their more subtle characteristics; the results are therefore used for identifying new criteria, thus enriching the criterion set.

The results of this evaluation highlight the strengths and weaknesses of each methodology and specify the features expected of APLE methodologies; thus, criteria-based evaluation provides a valuable framework for comparing the methodologies and selecting them according to specific project situations. Moreover, since the evaluation results pinpoint the shortcomings of methodologies, the evaluation framework can be used for improving current and future APLE methodologies [2], [4].

Another potential benefit of criteria-based evaluation is the applicability of the evaluation results in "Methodology Engineering": The results can be used as the basis for selecting and assembling reusable method chunks, instantiating abstract process frameworks, and extending existing methodologies in order to produce bespoke methodologies. This has been our ultimate intention in this research: We intend to develop a new APLE methodology by using methodology engineering methods; the target APLE methodology should make use of current APLE methodologies' desirable features while addressing their weaknesses. This requires identifying the various aspects of existing methodologies, which will be attained using the criteria-based evaluation results. It should be noted that in our research, we have studied and analyzed all of the existing APLE methodologies, but due to lack of space, only the most significant and well-documented methodologies will be focused upon in this paper.

The rest of this paper is organized as follows: Section 1 provides a brief review on eleven prominent APLE methods; Section 2 introduces the proposed evaluation criteria, based on which the evaluation results are presented and discussed in Section 3; and Section 4 provides the conclusions and suggests ways for furthering this research.

1. Review of APLE Methodologies

In this section, brief process-centered descriptions [5] are presented for the eleven APLE methodologies which are evaluated in later sections. The review focuses on the process, and the products/roles involved are mentioned as secondary to the process.

1.1. CDD (Component-Driven Development)

CDD utilizes the Feature-Driven Development (FDD) methodology [5] in order to combine PLE and agility. The reason for naming the methodology as CDD is that it shifts the focus from *features* to *components* [6]. CDD is not a full lifecycle methodology since it only encompasses Domain Engineering (DE), and even in this sub-process, it is only concerned with developing the Product Line (PL) architecture and core assets. This methodology consists of seven phases (as shown in Figure 1):

- Develop an Overall Model: Overall knowledge is acquired about the domain. Based on this knowledge, an informal list of features (and optionally, an object model) is developed, specifying the commonalities and variabilities.
- Build a Features List: PL features which manifest the functional and non-functional requirements are elicited and documented in a features list.
- Design SPL Architecture: The PL architecture is developed and evaluated.

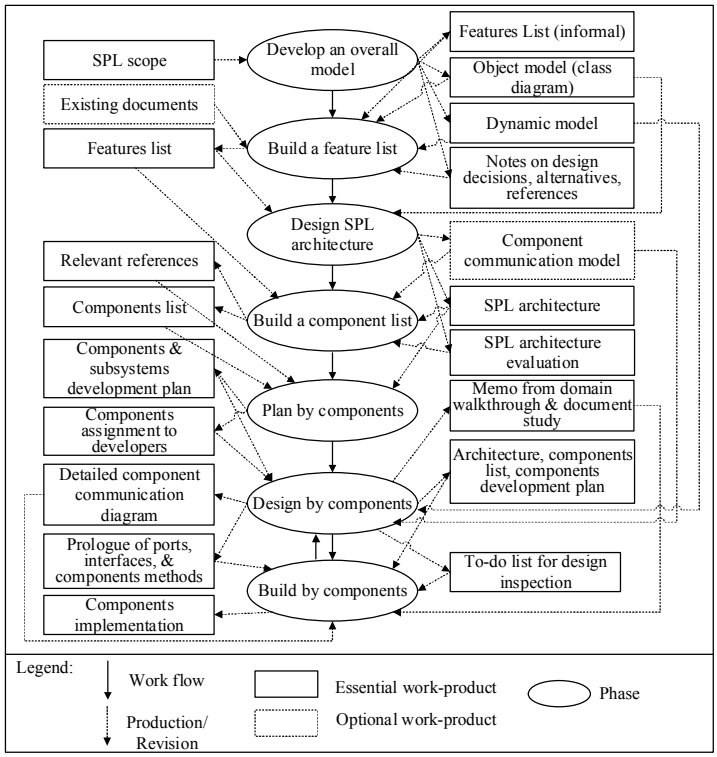

Figure 1. CDD process

- Build a Components List: A prioritized list is produced of the components identified while developing the architecture, as well as the components which are the result of decomposing the sub-systems. This list also shows the relationships among the components.
- Plan by Components: Based on the priorities, a development sequence is defined for the components, and components are assigned to the developers.
- Design by Components: After conducting a domain walkthrough and studying the available documents, detailed component communication diagrams are produced. This activity may result in a need to update the architecture, components list, and component development plan. Lastly, the ports, interfaces and method prologues of the components are produced.
- Build by Components: Components are implemented based on the design produced in the previous phase.

1.2. de Souza & Vilain

This method extends the Framework of Agile Practices (FAP) to propose an APLE process [7]. It encompasses both Domain Engineering (DE) (Figure 2) and Application Engineering (AE) (Figure 3); these sub-processes can be performed in tandem. The sub-processes and their constituent phases are as follows:

DE Sub-Process:

- Domain Analysis: PL applications' requirements are elicited and the PL feature model, demonstrating common and variable features, is developed.
- Domain Design: Considering the elicited features, the components and PL architecture are identified and designed.
- Iteration Definition: Components and their requirements are assigned to the iteration. Also, implementation tasks are identified and assigned to developers.
- Develop System Increment: Design and implementation is done for the components assigned to the current iteration.
- Validate Increment: Developers inspect the code for defects.

AE Sub-Process:

- Definition of Requirements: Product requirements are elicited and prioritized. Utilizing the domain feature model, the required components are selected from among the core assets. Also, the overall application model (an instance of the domain feature model with the product requirements added) is produced.
- Assign Requirements to Iterations: Considering the priorities, requirements are assigned to iterations and also to developers.
- Develop System Increment: Design, implementation, and integration are done for the requirements assigned to the current iteration.
- Validate Increment: Developers inspect each other's code for defects
- Integrate Increment: The implemented increment is integrated and reviewed to check the satisfaction of iteration requirements.
- Validate System: In addition to the usual validation activities, a concise document of the system is produced; the system is then delivered to the customer along with this document.

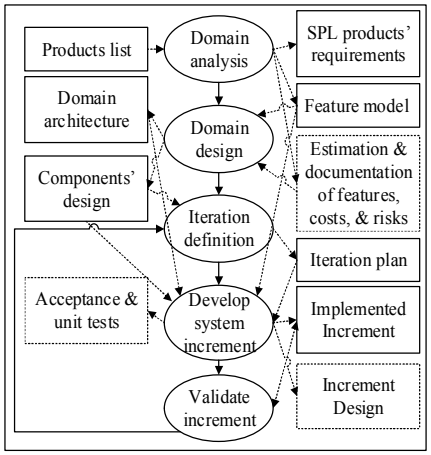

Figure 2. de Souza & Vilain's DE sub-process

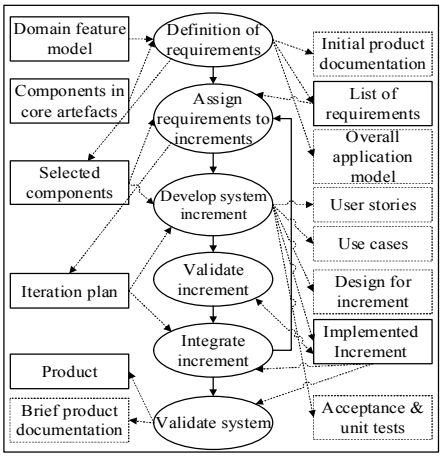

Figure 3. de Souza & Vilain's AE sub-process

1.3. RiPLE-SC

RiPLE-SC proposes an agile approach to PL scoping [8]. This process is a sub-process of a full-coverage PLE process called RiPLE (RiSE Process for Product Line Engineering). It encourages face-to-face conversation and meetings to enhance communication among developers. Phases are as follows (Figure 4):

- Pre-Scoping: Project vision, stakeholders, business goals, and organizational and operational context of the organization are identified, and the relevant markets are analyzed.
- Domain Scoping: Domains and sub-domains are analyzed, and the most relevant ones are identified.
- Product Scoping: Features of the domains are specified, based on which the relevant products are identified and documented in a product-map.
- Assets Scoping: Through the use of special metrics, the product-map is prioritized. Based on this prioritized product-map, the features which are most appropriate for reuse are identified and earmarked as assets.

1.4. A-Pro-PD

This methodology proposes an agile approach to product derivation (i.e. AE) in PLE [9]. The authors consider their proposed approach as a generic agile process model for product derivation in PLE. Phases are as follows (Figure 5):

- Preparing for Derivation: Product requirements are elicited, prioritized, and assigned to iterations.
- Product Configuration: According to product requirements and through reusing the available assets, a partial product configuration is developed.
- Product Development and Testing: The product parts which belong to the current iteration and which cannot be satisfied by reusing the core assets are developed and tested. The deployment of the product into the user environment is also carried out in this phase.

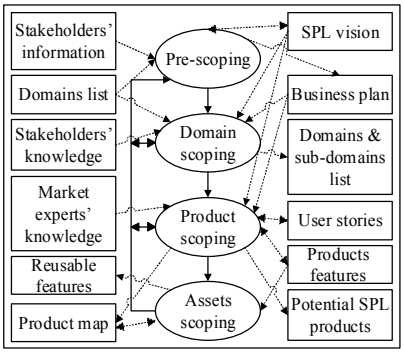

Figure 4. Process of RiPLE-SC

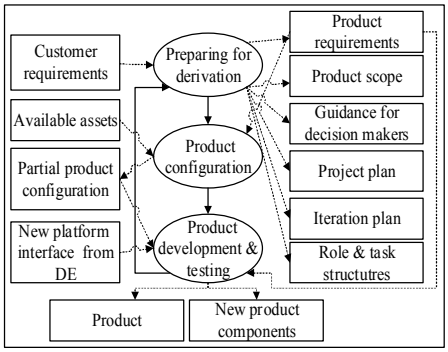

Figure 5. A-Pro-PD process

1.5. Díaz et al.

In the definition of this methodology, the three concepts of "Working Architecture", "Plastic Partial Components (PPCs)", and "Reflexive Reuse" are first introduced; based on these concepts and the process of the Scrum methodology [5], the proposed APLE process is defined [10]. The phases are as follows (Figure 6):

- Pregame: PL requirements are elicited and recorded in the SPL Backlog. These requirements are then translated into common and variable features, and are documented in the feature model.
- SPL Release Definition: Common and variable features are prioritized and divided into sprints (iterations), considering their priorities.
- Sprint Planning: Planning and estimation is performed for the features to be implemented in the current sprint. This information, which shows the sprint goal, is documented in the sprint backlog.
- Sprint-Domain Engineering: Parts of the feature model, core assets (using PPCs), and PL-architecture that belong to the current sprint are implemented.
- Sprint-Application Engineering: Parts of the product that belong to the current sprint are implemented by reconfiguring the PL-architecture and completing the partial implementations of PPCs.
- Review and Retrospective: Review meetings are held, and the collected feedback is relegated to the next sprint.

1.6. Ghanam & Maurer-2008

This methodology is targeted at agile organizations that aim to integrate PLE into their development processes [11]. In this methodology, acceptance tests (ATs) have a pivotal role, and the core assets are mined from products. It is assumed that the organization is an agile one which uses Test-Driven Development (TDD) techniques, and which has previously developed two separate systems in the same domain; the organization has now received requests for building a third system in the same domain. It has therefore decided to migrate to PLE. The proposed process is executed when the development of the third system is initiated (Figure 7), and is repeated for the development of each new system in the domain; the core assets are thus completed gradually. The phases of this methodology are as follows:

- Core Assets Team: A team, consisting of the senior developers of all of the three systems, is formed in order to mine the reusable modules.
- Evaluation and Extraction: Requirements for the third system are elicited and translated into ATs. These tests are compared to the ATs of previous systems. The output of this phase is a test model consisting of two layers: A generic layer encompassing the common parts among the compared tests, and a variability layer comprising the variable parts among these tests.
- Refactoring: The core-assets team refactors existing code to implement a module corresponding to the generic layer of the test model; development teams implement the modules corresponding to the test model's variable layer.
- Managing Core Assets: The refactored module is added to the core assets repository (referenced by its corresponding ATs) along with its variants.
- Core Asset Incorporation: The developed module is sent to the development team in order to be incorporated into the system under development.
- Architecture Evolution: While the core assets are being completed over time, the architecture is also evolved in a bottom-up fashion.

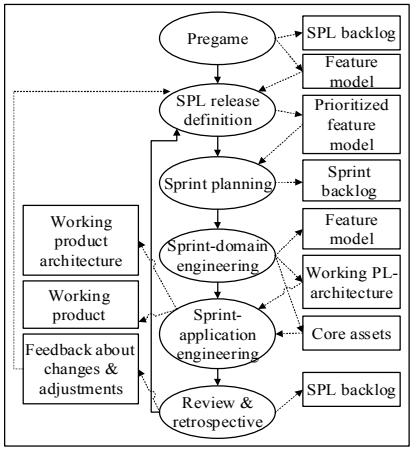

Figure 6. Diaz et al.'s process

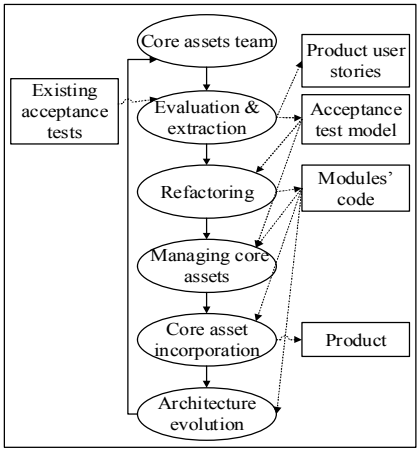

Figure 7. Process of Ghanam & Maurer-2008

1.7. Ghanam & Maurer-2009

This methodology proposes a PL instantiation process which is based on acceptance tests (ATs) [12]. It assumes that ATs and their corresponding code are available in the repository prior to running the process. The phases are as follows (Figure 8):

- Select ATs: Available ATs are presented to the customer, who selects a subset according to his/her requirements.
- Execute ATs: Selected ATs are executed using a test coverage tool so that the parts of the code that correspond to these tests can be identified.
- Extract Code: Based on the test coverage report, units of code that correspond to the selected ATs are extracted and formed into a new instance of the PL.
- Verify and Build: The instantiated software system is compiled, and the selected ATs are run against it to verify the satisfaction of user requirements.

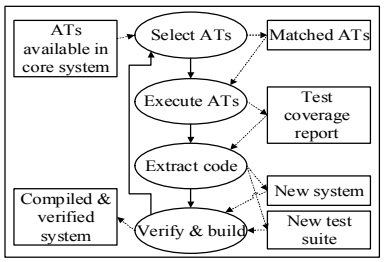

Figure 8. Process of Ghanam & Maurer-2009

After development, two extra activities are performed: 1) New customer requirements which could not be satisfied through reusing the core assets are identified, and for each new requirement, the corresponding AT and code are produced; and 2) PL maintenance is performed, typically resulting in changes to the code; when change occurs, all the instances which include the changed code unit should be re-instantiated and tested.

1.8. da Silva

The methodology proposed by da Silva is an agile method for PL scoping [13]. The phases of this process are as follows (Figure 9):

- Define Pre-Scoping: Candidate products or sub-domains are evaluated.
- Define Features: Features are elicited based on the previous phase's results.
- Analyze Commonality and Variability: The commonalities and variabilities among the pre-scoping results are analyzed and reflected onto the features.
- Release Scope: Features are prioritized and the effort required for their development is estimated; a subset of the features is then marked for release.
- Select Features for Implementation: Implementation iterations are defined and acceptance tests are produced.

1.9. Carbon et al.

The methodology proposed by Carbon et al. utilizes PuLSE-I (PuLSE-Instantiation), which is the product development part of the PuLSE methodology, and incorporates agility into it [14]. The phases are as follows (Figure 10):

- Plan for a Product Line Instance: Product requirements and PL scope are compared; if there is a requirement which is out of the PL scope, a request is sent to the family engineering (i.e. DE) team to extend the PL scope if advisable; otherwise, the requirement will be considered as a product-specific one. A project plan is produced containing effort estimates for the activities.
- Instantiate and Validate Product Line Model: The product specification (which shows the requirements) is built by instantiation from the product line model.
- Instantiate and Validate Reference Architecture: The product architecture is built through instantiation from the reference architecture.
- Construct Product: Detailed design, implementation, and testing are performed. This is done by developing the product-specific parts and also through reusing the components in the PL's assets base.
- Deliver System: The constructed system is deployed into the user environment.

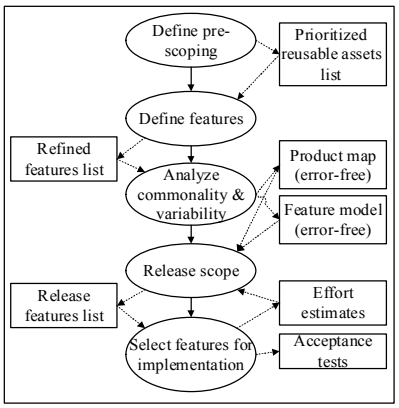

Figure 9. Process proposed by da Silva

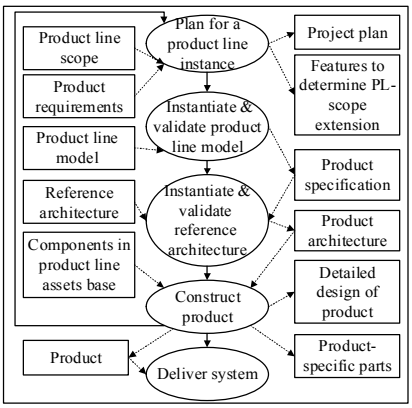

Figure 10. Process proposed by Carbon et al.

1.10. Noor et al.

This methodology proposes an agile approach to PL scoping [15], aimed at organizations that, after producing several similar products, have decided to construct a product line to improve efficiency. It promotes strong stakeholder collaboration by utilizing Collaboration Engineering (CE) patterns. Phases are as follows (Figure 11):

- Identify and Agree on Relevant Domains: Stakeholders review existing relevant domains, and agree on a list of domains to work on.
- Define Features for Each Domain: Stakeholders identify features for each domain; they could be new features, or features elicited from existing products.
- Discuss, Analyze, and Agree on Products: Considering the results of previous phases, stakeholders define PL products.
- Define Products in Terms of Features: Identified features and products are related and documented in the product map.
- Prioritize Product Map: The product map is prioritized based on the business value and feasibility of the features.

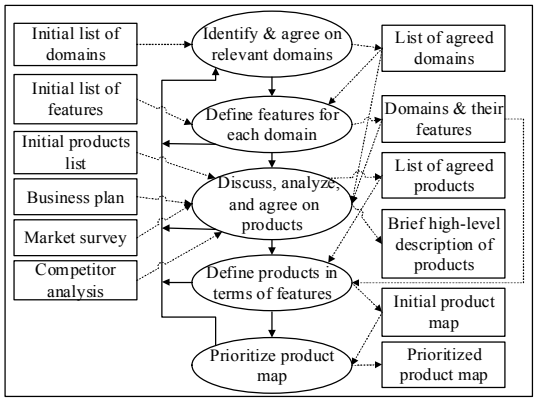

Figure 11. Process proposed by Noor et al.

1.11. Ghanam et al.

The methodology proposed by Ghanam et al. provides a method for variability management in agile organizations which intend to apply PLE to their development processes [16]. The proposed process will be executed when a new requirement arises. Phases are as follows (Figure 12):

- Eliciting New Requirements: New customer requirements are elicited and assigned to the iterations.
- Variability Analysis: Requirements are analyzed to determine commonalities and variation points among new and existing requirements.
- Updating the Variability Profile: Information gained during the previous phase is applied to the variability profile.
- Refactoring the Architecture: According to the changes that have been made to the variability profile, the architecture is refactored so that the new functionality can be implemented.
- Running the Tests: To make sure that the refactoring process has not had any adverse side effects, all the available tests are run.
- Realizing the New Requirements: New requirements that have caused the architecture to be refactored are implemented.
- Running the Tests (again): All tests (for new requirements and older ones) are run to make sure that requirements have been implemented properly and without negative consequences.

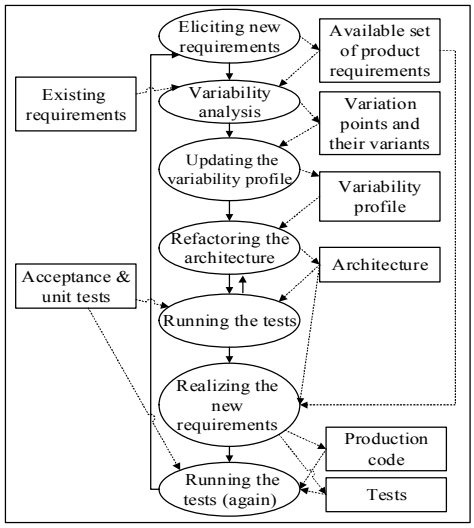

Figure 12. Process proposed by Ghanam et al.

2. Proposed Evaluation Criteria

As mentioned before, the evaluation presented herein is based on evaluation criteria. For the purpose of this evaluation, we have proposed a special set of criteria, using a

method similar to the approach introduced in [17]. Our criteria are divided into three categories according to the type of evaluation results obtained through applying them:

- Simple form (SM): The evaluation results for the criteria in this group are of the "Yes/No" type.
- Scale form (SC): The evaluation results for these criteria are enumerations, in that their value is chosen from among a limited and predefined set of categories (discrete values or levels).
- Descriptive form (D): The evaluation results for the criteria in this group are descriptive (narrative) statements.

In order to further manage the complexity of the criteria, we have also defined a separate, orthogonal categorization for the proposed criteria according to their semantics. To this aim, the proposed criteria have been divided into five separate groups: 1) General criteria for evaluating methodologies; 2) Criteria related to the characteristics of agile methods; 3) Criteria related to PLE characteristics; 4) Criteria related to the common goals of agile and PLE; and 5) Criteria related to issues arising due to the combination of the two approaches.

It is essential to demonstrate that the proposed criteria are valid. We have therefore strived to ensure that our proposed evaluation criteria satisfy the four validity meta-criteria of [18]. As a result, the proposed criteria are *general* enough to be applied to all APLE methodologies, *precise* enough to help discern the similarities and differences among APLE methodologies, *comprehensive* enough to cover all the important characteristics of APLE methodologies, and they are also *balanced*, in that they cover all the major types of features in a methodology: Technical, Managerial and Usage. The proposed criteria are explained in the following subsections.

2.1. General criteria for evaluating methodologies

There are certain characteristics that should be addressed by all software development methodologies, regardless of paradigm and context. The criteria used for evaluating these characteristics have been categorized under "general criteria", and are introduced in this subsection. The general criteria for evaluating software development methodologies have been divided into two categories: Criteria for evaluating the modeling language (Table 1) and criteria for evaluating the process (Table 2). This division is due to the fact that each of the two constituents of a methodology, the process and the modeling language, has its own specific set of concerns and features. Hence, each part is evaluated separately and according to its own characteristics.

Table 1. General criteria for evaluating methodologies – Modeling language group

Name	Description	Type	Possible values
Specific Modeling Language (ML)	Is a specific ML prescribed or enforced?	SC	1: Not prescribed/enforced; 2: Prescribed; 3: Enforced.
Simplicity to learn and use [19]	Is the ML simple to learn and use?	SM	Yes / No
Power of language [19]	Is the ML powerful enough (e.g., in support for various views and granularity levels)?	SM	Yes / No
Complexity management [19]	Does the ML support complexity management?	SM	Yes / No
Management of inconsistencies [19]	Does the ML provide mechanisms for handling inconsistencies in models?	SM	Yes / No

Table 2. General criteria for evaluating methodologies – Process group

Name		Description	Type	Possible values
Lifecycle				
Generic lifecycle coverage		Which phases of the generic development lifecycle are covered by the process?	SC	"D": Definition; "C": Construction; "M": Maintenance
Seamless transition [20]		Is the transition between phases seamless?	SC	1: No; 2: Potentially; 3: Yes
Smooth transition [20]		Is the transition between phases smooth?	SC	1: No; 2: Potentially; 3: Yes
Type of lifecycle		What is the type of the process lifecycle?	D	(e.g., waterfall, iterative-incremental)
Attention to design		Are design activities covered by the process?	SM	Yes/No
Integration with other methodologies [19]		Can the methodology be integrated with other methodologies (to address deficiencies)?	SC	Integration strategy: 1: Not required; 2: required but not provided; 3: provided.
Work-Products				
Adequacy [4]		Are the products related to each phase of the development process produced?	SC	Relevant products in: 1: No phases; 2: Some phases; 3: All phases.
Consistency [4]		Do the products complement each other with minimum overlap?	SC	1: Products overlap; 2: Products do not overlap.
Supported views [4]		Which views are supported by the work-products?	SC	"S": Structural; "F": Functional; "B": Behavioral
Granularity levels		Which granularity levels are supported by the work-products?	SC	"S": System; "P": Package; "C": Component; "O": Object **Or** "D": Domain; "SD": Sub-Domain; "PR": Product; "F": Features
Abstraction levels [4]		Which abstraction levels are supported by the work-products?	SC	"A": Analysis; "D": Design; "I": Implementation
Testability [20]		Are the products testable? The satisfying parameters include: Low number of products, understandability of products, and clarity of dependencies among products.	SC	Testability is: 1: weak (none of the parameters are satisfied); 2: average (some parameters are not satisfied); 3: strong (all parameters are satisfied).
Tangibility		Are the products tangible -clearly understandable- to their intended audience (customer and/or development team)? Products tangible to the customer include: requirements, analysis documents, and implemented system.	SC	1: Some products are not tangible to their intended audience; 2: Some products are not tangible to team members; 3: Some products are not tangible to the customer; 4: All products are tangible.
Traceability to reqs.		Are the products traceable to requirements?	SM	Yes/No
People				
Definition of roles		Are the involved roles defined along with their responsibilities?	SC	1: Roles not defined; 2: Roles defined, but without responsibilities; 3: Both roles and responsibilities defined.
Team knowledge & experience [19]		Is a specific type of knowledge, skill or experience required for team members?	SM	Yes/No
Team motivation mechanisms		Do any people management mechanisms exist to motivate team members?	SM	Yes/No
Usability				
Well-definedness	Expressiveness	Is the description of the methodology understandable and unambiguous?	SC	1: No; 2: To some extent; 3: Yes
	Completeness [19]	Which of the required definitions are provided by the methodology?	SC	"L": Lifecycle; "A": Activities; "TP": Techniques/Practices; "R": Roles; "P": Products; "U": Umbrella Activities; "RL": Rules; "ML": Modeling Language.

Table 2. General criteria for evaluating methodologies – Process group (Contd.)

Name		Description	Type	Possible values
Well-definedness (Contd.)	Rationality and consistency	Are the defined activities consistent with each other? Is their rationality evident?	SC	1: Problems in consistency & rationality; 2: Problems in consistency; 3: Problems in rationality; 4: No problems.
	Complexity management	Has the complexity of definition been managed (through hierarchical definition at phase-, stage-, and task levels)?	SM	Yes/No
	Attention to detail	How detailed are the definitions of tasks and phases?	SC	Details are provided for: 1: none of the phases; 2: some of the tasks or phases; 3: all the phases and their internal tasks.
	Definition of phase inputs and outputs (I/O)	Are the input- and output work-products (I/O) defined for the phases?	SC	1: I/O has not been defined; 2: I/O has been defined implicitly; 3: I/O has been explicitly defined for all phases.
Available resources	Available information [4]	Is there enough documentation and information available on the methodology?	SM	Yes/No
	Tool support [4]	Is there a CASE tool that supports this process?	SM	Yes/No
Practicality	Ease of use	Is the proposed process easy to use?	SC	1: Weak; 2: Average; 3: Good.
	Accounts of practical use [19]	Is there any evidence on the practical use of the methodology?	SM	Yes/No
Process manipulation	Configurability	Is the process configurable at the start of the project?	SC	1: No; 2: Possible, but not addressed explicitly; 3: Explicitly addressed
	Flexibility	Is the process reconfigurable while running the project?	SC	1: No; 2: Possible, but not addressed explicitly; 3: Explicitly addressed
Support for different project types	Criticality level	What Criticality Level (CL) can be addressed when using this methodology?	SC	CL: 1: has been defined explicitly (as specified); 2: has not been defined explicitly, but can be inferred (as specified); 3: has not been defined and cannot be inferred.
	Platform-adaptivity [4]	Is it possible to adjust the resources (existing tools and libraries) for a specific project?	SM	Yes/No
	Formalism	Are formal aspects supported in the process?	SM	Yes/No
	Scalability	Which sizes of projects are addressed?	SC	1: Small; 2: Medium; 3: Large.
Maintainability				
Modularity [4]		Is the product produced in a modular form?	SM	Yes/No
Requirements				
Reqs. elicitation		How are the requirements elicited?	D	
Reqs. specification [19]		What is the format for documenting the reqs.?	D	
Reqs.-based process [20]		Are the requirements elicited at the start of the process and used as a basis for development?	SM	Yes/No
Reqs. prioritization		Are the requirements prioritized?	SM	Yes/No
Application Constraints				
Constraints and/or assumptions		Are there any specific constraints/assumptions that should be observed (e.g., legal, technical, managerial, or geographical)?	SC	1: Constraints/Assumptions exist 2: Constraints/Assumptions prescribed 3: There are no constraints/assumptions

2.2. Criteria related to characteristics of agile methods

Since an APLE methodology should fulfill the goals and features common to all agile methods, we have defined specialized criteria to evaluate the degree of realization of these goals and features in APLE methodologies. These criteria are listed in Table 3.

Table 3. Criteria related to agility characteristics

Name		Description	Type	Possible values
Attention to Customer				
Early and continuous delivery of working software		Is the first version of software delivered early? Are further releases delivered continuously?	SC	Product is delivered: 1: neither early nor continuously; 2: continuously but not early; 3: early and continuously.
Active user involvement		Is the user directly involved in the process?	SM	Yes/No
Continuous customer feedback		Is customer feedback provided continuously?	SM	Yes/No
Teams				
Self-organizing teams		Are the teams self-organizing?	SC	Self-organization is: 1: Not discussed; 2: Addressed; 3: Ignored.
Face-to-face conversation		Has face-to-face communication of information been addressed?	SM	Yes/No
Velocity monitoring & control		Is the teams' velocity monitored so that they proceed at a sustainable pace?	SM	Yes/No
Attention to team behavior/ efficiency		Is the teams' performance and behavior monitored and tuned at regular intervals?	SM	Yes/No
Task assignment method		How are the tasks assigned in the process?	D	
Product				
Continuous integration		Is the software integrated continuously (at the end of iterations)?	SM	Yes/No
Modeling coverage		Are models included in the work-products?	SM	Yes/No
Standards		Is there any standard for producing the work-products (such as coding standard in XP [5])?	SM	Yes/No
Process				
Iterative-Incremental process		Is the software developed in an iterative - incremental fashion?	SM	Yes/No
Specific agile techniques		Are common agile techniques prescribed?	SM	Yes/No
Degree of agility (Based on the parameters introduced in [21])	Flexibility	Are expected and unexpected changes in requirements accommodated? NB: This is equivalent to the criteria in "Management of Changes in Reqs." subgroup of the criteria related to common goals of agile and PLE; so the evaluation will be done under that section.	SM	Yes/No
	Speed	Are the products produced rapidly?	SC	1: No; 2: To some extent; 3: Yes.
	Leanness	Are leanness factors addressed (short time spans, and the use of economical, simple, and high-quality tools)?	SM	Yes/No
	Learning	Is learning from previous iterations or previous projects addressed?	SC	1: Not addressed; 2: Addressed implicitly; 3: Addressed explicitly.
	Responsiveness	Is feedback provided by the methodology?	SM	Yes/No

2.3. Criteria related to PLE characteristics

In addition to agile features, PLE characteristics should also be addressed by APLE methodologies. Thus, we have defined a specialized set of criteria to evaluate the degree of realization of PLE characteristics in APLE methodologies (Table 4).

Table 4. Criteria related to PLE characteristics

Name	Description	Type	Possible values
Presence of PL-Specific Activities			
DE activities	Which DE-specific activities are addressed in the process?	SC	"S": Scoping; "A": Reference architecture; "CA": Core assets development.
AE activities	Which AE-specific activities are addressed in the process?	SC	"R": Matching product requirements & core requirements; "A": Reference architecture instantiation; "CA": Core assets selection; "V": Binding of variation points to variants; "P": product-specific parts development.
Product Line Characteristics			
Extensibility of PL scope	Is it possible to extend the scope of the PL?	SM	Yes/No
Reference architecture	Is the reference architecture produced and adhered to?	SC	1: Not produced; 2: Produced but not adhered to; 3: Produced and adhered to.
Techniques for Performing PL-Specific Activities			
Core assets identification	Is a method prescribed for identifying core assets & commonalities/variabilities (C/V)?	SM	Yes/No
Documenting C/V	Is a method prescribed for documenting C/V?	SM	Yes/No
Core assets selection	Is a method prescribed for selecting assets?	SM	Yes/No
Development of product-specific parts	Is a method prescribed for developing product-specific parts?	SM	Yes/No
Management			
Organization management	Is there any mechanism for organization management in PL?	SM	Yes/No
Core assets configuration management	Is a mechanism prescribed for configuration management of core assets?	SM	Yes/No

2.4. Criteria related to the common goals of agile development and PLE

This subsection will introduce the criteria that can evaluate the degree to which the common goals of PLE and agile development are fulfilled by APLE methodologies. This category has been defined because these goals are followed by both of the approaches which comprise agile product line engineering, and can hence be considered as the ultimate goals of APLE; therefore, the criteria that can evaluate the fulfillment of these goals in APLE methodologies have been grouped separately. These criteria have been explained in Table 5.

Table 5. Criteria related to the common goals of agile development and PLE

Name		Description	Type	Possible values
Increasing Customer Satisfaction				
On-time software delivery		Is reducing time-to-market addressed?	SM	Yes/No
Software quality	Core assets and software technical quality	Has attention been paid to technical quality (e.g., for design or code)?	SC	1: Not addressed; 2: Only for core assets; 3: Only for products; 4: For core assets & products.
	Continuous review/revision of DE/AE work-products	Are the work-products of DE and AE reviewed and revised continuously?	SC	1: Not addressed; 2: Only for DE work-products; 3: Only for AE work-products; 4: For both DE and AE work-products.
	Continuous testing of core assets and software	Are the core assets and the product tested continuously?	SC	1: Not addressed; 2: For core assets; 3: For product; 4: For both core assets and product.
Efficiency				
Management of human resources		Has management of human resources (efficient employment of human assets) been addressed?	SM	Yes/No
Increasing temporal efficiency		Is a mechanism prescribed for increasing temporal efficiency?	SM	Yes/No
Management of Changes in Requirements				
Expected changes		Are expected changes in requirements accommodated?	SM	Yes/No
Unexpected changes		Are unexpected changes in requirements accommodated?	SM	Yes/No

2.5. Criteria related to the combination of agile development and PLE

These criteria are related to the issues that arise as the result of combining agile development and PLE (Table 6). It should be noted that these criteria are different from the criteria presented in Section 2.4, in that the criteria in Section 2.4 are intended to evaluate the fulfillment of the goals that are pursued by both agile and PLE methodologies, whereas the criteria in this section are related to the concerns that arise when agility and PLE are combined; for instance, the reuse approach might well be changed when agility is fused into a PLE approach.

Table 6. Criteria related to the issues arising when combining agile development and PLE

Name	Description	Type	Possible values
Basis of the methodology	What is the basis of the proposed process?	SC	PL; or Agile
Reuse approach	What is the reuse approach after combination?	SC	"P": Proactive; "R": Reactive; "RX": Reflexive [10]

3. Evaluation Results

In this section, we provide the results of evaluating the reviewed methodologies based on the proposed criteria. It should be noted that in the evaluation tables, "N/A" denotes "Not relevant to the context or properties of the methodology", and "N/D" signifies "Not defined in the methodology". The results of evaluating the methodologies based on the general criteria are given in Table 7 (Modeling Language) and Table 8 (Process).

Table 7. Evaluation results for general evaluation criteria – Modeling language group

Criteria \ Methodologies	CDD [6]	de Souza & Vilain [7]	RiPLE-SC [8]	Diaz et al. [10]	A-Pro-PD [9]	Ghanam & Maurer 2008 [11]	Ghanam & Maurer 2009 [12]	Ghanam et al. [16]	da Silva [13]	Carbon et al. [14]	Noor et al. [15]
Specific Modeling Language	2	2	1	1	1	1	1	1	1	1	1
Simplicity to learn and use	Yes	Yes	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Power of language	Yes	Yes	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Complexity management	Yes	No	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Management of inconsistencies	No	No	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 8. Evaluation results for general evaluation criteria – Process group

Criteria \ Methodologies	CDD [6]	de Souza & Vilain [7]	RiPLE-SC [8]	Diaz et al. [10]	A-Pro-PD [9]	Ghanam & Maurer 2008 [11]	Ghanam & Maurer 2009 [12]	Ghanam et al. [16]	da Silva [13]	Carbon et al. [14]	Noor et al. [15]
Lifecycle											
Generic lifecycle coverage	D-C	D-C	D	D-C	D-C	D-C	D-C	D-C	D	C	D
Seamless transition	3	2	3	1	1	1	1	1	3	2	3
Smooth transition	3	3	3	3	3	1	3	3	3	3	3
Type of lifecycle	Iter.-Incr.	Iter.-Incr.	Iter.-Incr.	Iter.-Incr.	Iter.-Incr.	Iter.-Incr.	Iter.-Incr.	Iter.-Incr.	Iter.-Incr.	Iter.-Incr.	Iter.-Incr.
Attention to design	Yes	Yes	N/A	Yes	No	No	No	No	N/A	Yes	N/A
Integration with other methodologies	2	1	3	1	2	1	2	1	2	3	2
Work-Products											
Adequacy	3	3	3	2	2	2	3	2	3	3	3
Consistency	2	1	2	1	2	2	2	1	2	2	2
Supported views	S-F-B	DE: S-F, AE: F	F	S-F	F	F	F	F	F	S-F	F
Granularity levels	S-P-C-O	DE: S-P-C-O, AE: N/D	D-SD-PR-F	S-P-C	N/A	N/A	N/A	N/A	SD-PR-F	S-P-C	D-PR-F
Abstraction levels	A-D-I	A-D-I	A	A-D-I	A-I	A-I	A-I	A-I	A	D-I	A
Testability	2	3	3	3	3	3	3	3	2	2	2
Tangibility	4	2	4	4	4	2	4	2	4	4	4
Traceability to requirements	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
People											
Definition of roles	3	1	3	1	1	1	1	1	1	1	3
Team knowledge & experience	Yes	No	Yes	No	No	Yes	Yes	No	No	No	Yes
Team motivation mechanisms	2	2	2	1	1	1	1	1	1	1	2

Table 8. Evaluation results for general evaluation criteria – Process group (Contd.)

Criteria \ Methodologies		CDD [6]	de Souza & Vilain [7]	RiPLE-SC [8]	Diaz et al. [10]	A-Pro-PD [9]	Ghanam & Maurer 2008 [11]	Ghanam & Maurer 2009 [12]	Ghanam et al. [16]	da Silva [13]	Carbon et al. [14]	Noor et al. [15]
Usability												
Well-definedness	Expressiveness	3	2	2	2	2	3	2	3	2	2	2
	Completeness	L-A-TP-R-P-U-RL-ML	L-A (Partial)-PT-P-U (Partial)-RL	L-A-PT-R-P-U (Partial)-RL	L-A-P-U (Partial)	L-A-PT-U (Partial)-P	L-A-PT-P	L-A-PT-P	L-A-PT-P	L-A-PT-U (Partial)-P	L-A-PT-U (Partial)-P	L-A-PT-R-U (Partial)-P
	Rationality & consistency	4	4	4	4	3	4	4	4	4	4	4
	Complexity management	Yes	Yes	Yes	No	No	No	No	No	No	No	No
	Attention to detail	3	2	3	1	1	2	3	3	2	2	3
	Definition of phase I/O	3	2	2	2	2	2	2	2	3	2	3
Available resources	Available information	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Tool support	No	No	No	No	No	No	No	No	No	No	No
Practicality	Ease of use	3	2	2	2	2	2	2	2	2	3	2
	Accounts of practical use	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	Yes	Yes
Process manipulation	Configurability	2	2	1	1	2	1	1	1	1	2	2
	Flexibility	2	2	1	1	1	1	1	1	1	1	2
Different project types	Criticality level	3	3	3	3	3	3	3	3	3	3	3
	Platform-adaptivity	N/A	N/A	N/A	Yes	N/A	N/A	N/A	N/A	N/A	N/A	Yes
	Formalism	No	No	No	No	No	No	No	No	No	No	No
	Scalability	3,2	3,2,1	2	1	1	1	1	1	2,1	3,2,1	2,1
Maintainability												
Modularity		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Requirements												
Requirements elicitation		From domain experts, studying documents	DE: Analysis of past, current, and to-be systems, AE: Through user stories	Analysis of past systems, Through user stories	Through user stories	Talk to users and core assets team	Through user stories	Through selection of acceptance tests	Through user stories	Interview with domain expert, exploring past documents	N/D	Brain storming sessions among stakeholders
Requirements specification		Features List	DE: N/D , AE: User stories or use cases	User stories, Features	User stories, Feature Model	N/D	User Stories, Acceptance Tests	Acceptance Tests	User Stories	Feature Model	Product Specification	Features List
Requirements-based process		Yes	Yes	N/A	Yes	Yes	Yes	Yes	Yes	N/A	Yes	N/A
Requirements prioritization		Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes
Application Constraints												
Constraints and/or assumptions		1	2	1	1	1	1	1	1	1	1	1

The results of evaluating the reviewed methodologies by applying the criteria related to agile characteristics are presented in Table 9, and the results of evaluating the methodologies based on the criteria related to PLE characteristics are shown in Table 10.

Table 9. Evaluation results for criteria related to agility characteristics

Criteria \ Methodologies		CDD [6]	de Souza & Vilain [7]	RiPLE-SC [8]	Diaz et al. [10]	A-Pro-PD [9]	Ghanam & Maurer 2008 [11]	Ghanam & Maurer 2009 [12]	Ghanam et al. [16]	da Silva [13]	Carbon et al. [14]	Noor et al. [15]
Attention to Customer												
Early and continuous delivery		N/A	1	N/A	3	3	1	1	N/A	N/A	3	N/A
Active user involvement		Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
Continuous customer feedback		No	No	Yes	Yes	Yes	No	Yes	Yes	No	Yes	No
Teams												
Self-organizing teams		2	3	1	1	1	1	1	1	1	1	3
Face-to-face conversation		No	Yes	Yes	Yes	No	Yes	No	No	Yes	Yes	Yes
Velocity monitoring & control		No	No	No	No	No	No	No	No	Yes	No	No
Attention to team behavior/efficiency		No	No	Yes	No	No	No	No	No	No	No	Yes
Task assignment method		Components assigned by chief programmers	Reqs. assigned at the start of iterations	Profile analysis	N/D	N/D	N/D	N/D	N/D	N/D	N/D	N/D
Product												
Continuous integration		Yes	Yes	N/A	No	Yes	No	Yes	Yes	N/A	Yes	N/A
Modeling coverage		Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes
Standards		Yes	No	Yes	No	No	No	No	No	No	No	No
Process												
Iterative-Incremental Process		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Specific agile techniques		Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Degree of agility	Speed	2	3	2	3	3	2	3	3	2	2	2
	Leanness	Yes	Yes	No	Yes	No	No	No	Yes	No	No	Yes
	Learning	2	3	1	1	1	1	1	1	2	2	1
	Responsiveness	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes

Table 10. Evaluation results for criteria related to PLE characteristics

Criteria \ Methodologies	CDD [6]	de Souza & Vilain [7]	RiPLE-SC [8]	Diaz et al. [10]	A-Pro-PD [9]	Ghanam & Maurer 2008 [11]	Ghanam & Maurer 2009 [12]	Ghanam et al. [16]	da Silva [13]	Carbon et al. [14]	Noor et al. [15]
Presence of PL- Specific Activities											
DE activities	A-CA	A-CA	S	A-CA	N/A	A-CA	N/A	A-CA	S	N/A	S
AE activities	N/A	R-CA	N/A	A-CA-V	CA-P	R-CA	R-CA-P	N/A	N/A	R-A-CA-V-P	N/A
Product Line Characteristics											
Extensibility of PL scope	N/A	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Reference architecture	2	2	N/A	3	1	1	1	3	N/A	3	N/A
Techniques for Performing PL-Specific Activities											
Core assets identification	No	No	Yes	No	N/A	Yes	N/A	Yes	Yes	N/A	Yes
Documenting commonalities/variabilities	Yes	Yes	Yes	Yes	N/A	Yes	N/A	Yes	Yes	N/A	No
Core assets selection	N/A	Yes	N/A	No	No	Yes	Yes	N/A	N/A	Yes	N/A
Development of product-specific parts	N/A	Yes	N/A	No	No	No	Yes	N/A	N/A	No	N/A
Management											
Organization management	No	No	Yes	No	Yes	No	No	No	Yes	No	Yes
Core assets configuration management	Yes	Yes	N/A	Yes	N/A	Yes	Yes	Yes	N/A	N/A	N/A

Table 11 shows the results of evaluation based on the criteria related to common goals of agile development and PLE, and Table 12 contains the results of evaluation based on the criteria related to issues that arise when combining agility and PLE.

Table 11. Evaluation results for criteria related to the common goals of agile development and PLE

Criteria \ Methodologies		CDD [6]	de Souza & Vilain [7]	RiPLE-SC [8]	Diaz et al. [10]	A-Pro-PD [9]	Ghanam & Maurer 2008 [11]	Ghanam & Maurer 2009 [12]	Ghanam et al. [16]	da Silva [13]	Carbon et al. [14]	Noor et al. [15]
Increasing Customer Satisfaction												
On-time delivery of software		N/A	Yes	N/A	Yes	Yes	Yes	Yes	N/A	N/A	Yes	N/A
Software quality	Core assets and software technical quality	2	4	N/A	1	4	1	1	1	N/A	1	N/A
	Continuous review/revision of DE/AE work-products	2	4	2	4	3	4	3	2	2	3	2
	Continuous testing of core assets and software	1	4	N/A	1	3	4	3	2	N/A	3	N/A
Efficiency												
Management of human resources		Yes	No	Yes	No	No	No	No	No	No	No	Yes
Increasing temporal efficiency		No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes

Table 11. Evaluation results for criteria related to the common goals of agile development and PLE (Contd.)

Criteria \ Methodologies	CDD [6]	de Souza & Vilain [7]	RiPLE-SC [8]	Diaz et al. [10]	A-Pro-PD [9]	Ghanam & Maurer 2008 [11]	Ghanam & Maurer 2009 [12]	Ghanam et al. [16]	da Silva [13]	Carbon et al. [14]	Noor et al. [15]
Management of Changes in Requirements											
Expected changes	Yes	Yes	Yes	Yes	N/A	Yes	N/A	Yes	Yes	N/A	No
Unexpected changes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table 12. Evaluation results for criteria related to the issues arising when combining agility and PLE

Criteria \ Methodologies	CDD [6]	de Souza & Vilain [7]	RiPLE-SC [8]	Diaz et al. [10]	A-Pro-PD [9]	Ghanam & Maurer 2008 [11]	Ghanam & Maurer 2009 [12]	Ghanam et al. [16]	da Silva [13]	Carbon et al. [14]	Noor et al. [15]
Basis of the methodology	Agile	PL	PL	Agile	PL	Agile	Agile	Agile	PL	PL	PL
Reuse approach	N/A	P-R	N/A	RX	P-R	R	R	R	N/A	R	N/A

4. Conclusions and Future Work

The development of a proper APLE methodology which combines agility with PLE effectively, and which embodies the features of both of these successful approaches, would be considered a significant achievement in software engineering. This is due to the fact that without a well-defined methodology, it will not be possible to effectively apply APLE to real projects. This has been our ultimate goal in this research: To propose an effective and practical APLE methodology which significantly improves on the status quo. As the first step to this goal, we have analyzed and evaluated existing APLE methodologies using a criteria-based approach, the main results of which have been reported in this paper. The evaluation results reveal that there is no single methodology which covers all of the following APLE features (which are considered desirable, or even essential, in APLE methodologies): Full coverage of the generic software development lifecycle, comprehensive and precise definition of the methodology, sufficient attention to umbrella activities, prescription of a specific modeling language, provision of model examples, attention to learning (at project- and portfolio levels), attention to active user involvement, and management of expected and unexpected changes.

We aim to further this research by engineering an APLE methodology based on the evaluation results reported herein. This methodology will address the deficiencies identified in current methodologies, while making use of their merits. The methodology will be continuously validated against the proposed criteria to ensure that all the features expected to be present in APLE methodologies are indeed implemented in the methodology produced. The target methodology will also be validated through enactment in an industrial-scale APLE project to demonstrate its practical efficacy.

References

[1] G. K. Hanssen, and T. E. Fægri, Process fusion: An industrial case study on agile software product line engineering, Journal of Systems and Software, vol. 81, no. 6, 2008, pp. 843–854.

[2] M. Asadi, and R. Ramsin, MDA-Based Methodologies: An Analytical Survey, in Proceedings of European Conference on Model Driven Architecture – Foundations and Applications, 2008, pp. 419–431.

[3] J. Díaz, J. Pérez, P. P. Alarcón, and J. Garbajosa, Agile product line engineering—A systematic literature review, Software: Practice and Experience, vol. 41, no. 8, 2011, pp. 921–941.

[4] S. Hesari, H. Mashayekhi, and R. Ramsin, Towards a General Framework for Evaluating Software Development Methodologies, in Proceedings of Computer Software and Applications Conference, 2010, pp. 208–217.

[5] R. Ramsin, and R. F. Paige, Process-centered Review of Object Oriented Software Development Methodologies, ACM Computing Surveys, vol. 40, no. 1, 2008, pp. 3:1–3:89.

[6] X. Wang, Towards an Agile Method for Building Software Product Lines, M.Sc. Thesis, University of York, UK, 2005.

[7] D. S. de Souza, and P. Vilain, Selecting Agile Practices for Developing Software Product Lines, in Proceedings of International Conference on Software Engineering & Knowledge Engineering, 2013, pp. 220–225.

[8] M. Balbino, E. S. de Almeida, and S. R. de Lemos Meira, An Agile Scoping Process for Software Product Lines, in Proceedings of International Conference on Software Engineering & Knowledge Engineering, 2011, pp. 717–722.

[9] P. O'Leary, F. McCaffery, S. Thiel, and I. Richardson, An Agile Process Model for Product Derivation in Software Product Line Engineering, Journal of Software: Evolution and Process, vol. 24, no. 5, 2012, pp. 561–571.

[10] J. Díaz Fernández, J. Pérez Benedí, A. Yagüe Panadero, and J. Garbajosa Sopeña, Tailoring the Scrum Development Process to Address Agile Product Line Engineering, in Proceedings of Jornadas de Ingeniería del Software y base de Datos, 2011.

[11] Y. Ghanam, and F. Maurer, An Iterative Model for Agile Product Line Engineering, in Proceedings of International Software Product Line Conference Doctoral Symposium, 2008, pp. 377–384.

[12] Y. Ghanam, and F. Maurer, Extreme Product Line Engineering: Managing Variability and Traceability via Executable Specifications, in Proceedings of Agile Conference, 2009, pp. 41–48.

[13] I. F. da Silva, An Agile Approach for Software Product Lines Scoping, in Proceedings of International Software Product Line Conference, 2012, pp. 225–228.

[14] R. Carbon, M. Lindvall, D. Muthig, and P. Costa, Integrating Product Line Engineering and Agile Methods: Flexible Design Up-front vs. Incremental Design, in Proceedings of International Workshop on Agile Product Line Engineering, collocated with International Software Product Line Conference, 2006, pp. 1–8.

[15] M. A. Noor, R. Rabiser, and P. Grünbacher, Agile product line planning: A collaborative approach and a case study, Journal of Systems and Software, vol. 81, no. 6, 2008, pp. 868–882.

[16] Y. Ghanam, D. Andreychuk, and F. Maurer, Reactive Variability Management in Agile Software Development, in Proceedings of Agile Conference, 2010, pp. 27–34.

[17] B. Kitchenham, S. Linkman, and D. Law, DESMET: a methodology for evaluating software engineering methods and tools, Computing & Control Engineering Journal, vol. 8, no. 3, 1997, pp. 120–126.

[18] G. M. Karam, and R. S. Casselman, A Cataloging Framework for Software Development Methods, Computer, vol. 26, no. 2, 1993, pp. 34–45.

[19] M. Taromirad, and R. Ramsin, CEFAM: Comprehensive Evaluation Framework for Agile Methodologies, in Proceedings of Annual IEEE Software Engineering Workshop, 2008, pp. 195–204.

[20] R. Ramsin, and R. F. Paige, Iterative criteria-based approach to engineering the requirements of software development methodologies, IET Software, vol. 4, no. 2, 2010, pp. 91–104.

[21] A. Qumer, and B. Henderson-Sellers, Comparative evaluation of XP and Scrum using the 4D Analytical Tool (4-DAT), in Proceedings of the European and Mediterranean Conference on Information Systems, 2006, pp. 1–8.