Assessing the Adoption Level of Agile Development Within Software Product Lines: The AgiPL-AM Model

Hassan Haidar, Manuel Kolp, Yves Wautelet

Assessing the adoption level of agile development within Software Product Lines: the AgiPL-AM model

Hassan Haidar[1], Manuel Kolp[1], and Yves Wautelet[2]

[1] LouRIM-CEMIS, UCLouvain, Louvain-la-Neuve, Belgium
[2] KULeuven, Faculty of Economics and Business, Brussels, Belgium
hassan.haidar@uclouvain.be, manuel.kolp@uclouvain.be,
yves.wautelet@kuleuven.be

Abstract. Agile Product Lines are combinations of agile and product-line techniques. Introducing agile software development methods into software product lines makes the development processes evolve from predictive to iterative and incremental and offers flexibility to react on customers' changing requirements and market demand and deliver high quality software. However, this combination is still challenging and the maturity of an agile adoption is often hard to determine. Assessing the current situation regarding the combination is thus an essential step towards a successful integration of agile methods into software product lines. Following a specific research approach, we have built an assessment model called AgiPL-AM allowing self-evaluations within the team in order to determine the current state of agile software development in combination with software product lines. AgiPL-AM, our model for assessing organizational agility of Agile Product Line approaches, is comprised of six categories (five are related to agile principles and one to product line architecture) and five levels of maturity. The assessment results demonstrate that AgiPL-AM has the ability to reveal and pinpoint agile product-line approach strengths and weaknesses. It makes recommendations to improve the status and may give a guideline for this improvement.

Keywords: Agile Product Line Engineering, Agile Software Development, Process Maturity Model, Agile Assessment Model.

1 Introduction

To deal with the growing complexity of information systems and to handle the competitive and changing needs of the IT production industry, practitioners and researchers have proposed several approaches with the intention to combine agile and product lines techniques [1]. The goal was to make software product-line methodologies evolve from predictive to iterative and incremental, and to agile approaches.

Many questions could be asked about the conducted combinations, their results, and their effectiveness. If "agility" is considered as a "quality attribute" of the development process, two crucial research questions arise: *"how to combine agile practices*

with product-line techniques?".and *"how to assess the agility attribute of an agile product line method?"*.

This paper focuses on the second one and proposes an assessment model to determine the current state of agile development in combination with software product lines. In fact, assessing the status of the development is a crucial step for a successful combination of agile methods and software product lines. Thus, through this paper we propose an assessment model called AgiPL-AM that allows self-assessments within the "*domain* and *application* teams" in order to determine the current state of agile software development in the context of software product lines.

To develop our targeted assessment model we followed a research process of three phases. The first phase reviews the literature on maturity models that concern *Software Product Line Engineering*, *Agile Product Lines*, and *Agile Software Development*. The desired assessment model is built in the second phase. Finally, the third phase applies and evaluates the proposed model.

The obtained model (i.e. AgiPL-AM) is an agility assessment model for assessing agility of Agile Product Line approaches that comprises six categories (i.e. five categories are related to agile principles and one category is related to product line architecture) and five levels of maturity. To build the assessment model, we took an existing agile maturity model (SAMI model [14]) as a basis.

2 Background

This section consists of three parts. The first part introduces the combination of agile software development and software product lines. The second one presents existing assessment models that focus on software reuse strategies. The third part presents existing assessment models that focus on agile practices adoption.

2.1 Agile Software Product Lines

Research works such as [2, 3, 4, and 5] have demonstrated the difficulty of integrating agile methods with product line engineering due to the plan-driven and sequential nature of product line approaches versus the iterative and flexible nature of agile frameworks. However, they have highlighted that adding agility to product line engineering is not only possible but can also be highly beneficial [2].

Due to their actual benefits, agile methods could help product line teams to deal with the highlighted issue and thus being agile. According to [6], combining agile with Software Product Lines is not trivial, and thus, Agile Software Product Line Engineering has been identified as driven by an assumed improvement of customer collaboration and software development. It promises to deliver high-quality software at the required faster pace.

In practice, companies who intend to adopt an agile software product-line approach, assume that the development could benefit from both a working reuse strategy and an increased flexibility with the adopted agile practices. Note that this flexibility

is necessary to react on customer needs and changing requirements during the development process [6, 7].

Generally, in most cases, Software Product Line approaches are already used and companies target to transform towards agile [7]. Therefore, companies need approaches that integrate the agility while preserving the software product lines. Many already proposed models and approaches ensure the agility integration within software product lines and consequently help teams and companies to achieve their aim of being agile.

In the literature, several concrete approaches and methods that combine agile with product line concepts are available. For example, Tian and Cooper [2] mention two possible approaches : one approach is to take an existing SPL process and introduce agility; the other approach starts with an agile process and tailors it for SPLs. They identify the way to end up with a combination of Agility and Software Product Line Engineering. However, they do not give any recommendations on how to reach this state. In addition, dos Santos and Lucena [8], introduce the ScrumPL approach, which supports iterative domain and application engineering based on Scrum.

The review of the relevant agile software product line approaches shows that most of these approaches present only benefits after a successful agility integration and give a combination model. However, some of these approaches do not give any recommendations on how to reach the presented state. In addition, some of the reviewed approaches require suitable tools and appropriate infrastructure as a precondition to the successful integration of agile. Moreover, some approaches impediment during early phases of the agile adoption that are related to project management, coordination, and communication [7].

2.2 Assessment models for software reuse strategies

Over the past years, different assessment models were proposed to assess software reuse. Hereafter, we present three of them. The CMMI-DEV model [9] provides a collection of best practices that support organizations to improve their processes. It focuses on activities for developing products to meet needs of customers and a well-known standard that defines methods to evaluate complete process models and organizations.

Based on CMMI, Jasmine and Vasantha [10] have defined the Reuse Capability Maturity Model (RCMM). RCMM model focuses on a well-planned and controlled reuse oriented software development. This model comprises maturity levels that denote an achieved level in the evolution to a mature reuse process.

The “VDA QMC Working Group” has proposed the Automotive SPICE model [11]. that is a process assessment model that contains a set of indicators to be considered when interpreting the intent of the Automotive SPICE process reference model. These indicators may also be used when implementing a process improvement program subsequent to an assessment.

2.3 Assessment models for agile development

In practice, organizations are unable to fully adopt agile development practices immediately or over a short period since it requires a socio-technical transformation/migration process [12]. Accordingly, maturity models can help and guide organizations in providing the directions concerning the practices and the manner that they can be introduced and established in the organization.

Schweigert et al. [13] have identified several maturity models for agile development. They use a set of assessment criteria to assess each identified maturity model in term of their fitness of purpose, completeness, definition of agile levels, objectivity, correctness, and consistency. With these assessment criteria, they surveyed the related issues (e.g. *Whether the emphasis of the model is on assessing agile practices or not (i.e. Fitness of purpose)*). Following these criteria, they have concluded that the SAMI model (Sidky Agile Measurement Index) proposed by Sidky et al. [14] has the highest scores among the studied models. SAMI consists of two components. The first one is *an agile measurement index* and the second one is *a four-stage process*. Together, these two components guide and assist the agile adoption efforts of organizations.

SAMI is structured into four main parts: agile levels, agile principles, agile practices and concepts, and indicators. Driven from the values and principles of the "Agile Manifesto" [15], the model defines five agile levels:

- **Level 1**: is dedicated for the *Collaboration* which is one of the essential values and qualities of agile;
- **Level 2**: represents the objective of *"Developing software through an evolutionary approach"*;
- **Level 3**: represents the objective of *"Effectiveness and efficiency in developing high quality software"*;
- **Level 4**: is depicted for the objective of *"gaining the capability to respond to change through multiple levels of feedback"*;
- **Level 5**: represents the objective of *"Establishing a vibrant and all-encompassing environment to sustain agility"*.

In addition, the SAMI model has clustered the 12 agile principles into five categories that group the agile practices. These categories are: (1) Embracing change to deliver customer value; (2) Plan and deliver software frequently; (3) Human-centricity; (4) Technical excellence; (5) Customer collaboration. In total, SAMI incorporates 40 agile practices. Organization should start adopting agile practices on lower levels first, because the agile practices on a higher level are dependent on the practices introduced at the lower levels [14]. Moreover, Sidky et al. [14] have proposed a range of indicators that are used to assess certain characteristics of an organization or project, such as people, culture, and environment, in order to ascertain the readiness of the organization or project to adopt an agile practice [13]. The SAMI model contains about 300 different indicators for the 40 agile practices [17].

3 Research approach

To reach the main target of this paper, we have inspired our research procedure from the work of Hevner et al. [18]. Since our primary objective is to propose an assess-

ment artifact that could be used to assess the adoption level of agile devel-opment within Agile Software Product Lines, the followed research method in-volves mainly the three phases. The first phase is dedicated for the definition of the problem and the objectives of the assessment artifact. The second phase is devoted for the design and the development of the targeted assessment artifact. The third phase illustrates the applicability of the proposed model.

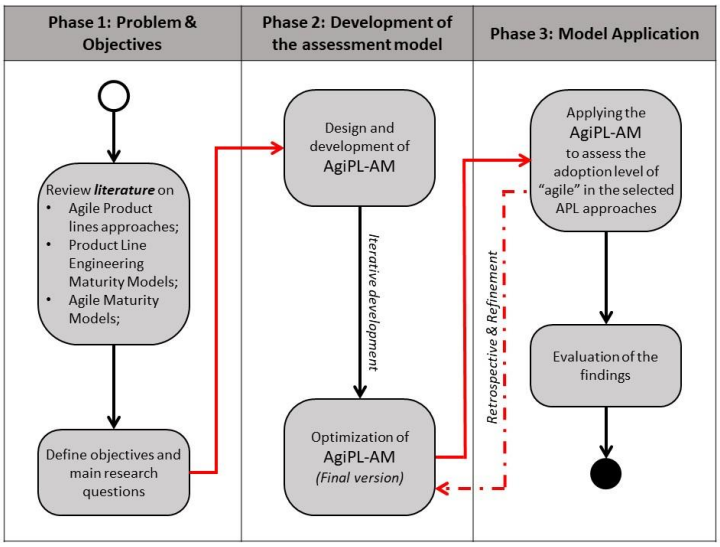

Fig. 1. Research procedure

Figure 1 depicts a detailed view on the procedure that we followed in order to de-velop, apply, and evaluate the Agility Assessment Model (i.e. AgiPL-AM) proposed in this paper. The first phase starts by a review of literature on maturity models that concern Software Product Line Engineering, Agile Product Lines, and Agile Software Development. The second phase involves the construction of the proposed "agile assessment model". After defining the main objectives of the required assessment model, the AgiPL-AM was designed and developed in an iterative way. In the third phase, the model was applied and evaluated. At this stage, the model was reviewed and refined in order to optimize and finalize AgiPL-AM.

4 Assessment Model for Agile Product Lines: AgiPL-AM

In order to attempt our target, we started by performing an extensive review of agile product line approaches, relevant case studies, and software process oriented-maturity models with emphasis on the agile software development approaches and on the agile product line approaches. It was identified that seven important areas need to be con-

sidered in an assessment for the combination of agile development and software product lines. According to Hohl et al. [22], these areas are the following:

1. *Product Line Architecture:* the objective of this area is to provide a suitable software architecture to enable the implementation of several software variants for different products with a high degree of software reuse;
2. *Domain Requirements Engineering:* behind this area, the purpose is to identify the reuse assets that should be developed in a software product line. This includes the identification of products and features that should be part of the product line and the definition of common and variable features;
3. *Agile Software Development:* the main target of this area is to react faster on customer needs and legal constraints to reduce the time-to-market for innovative feature upon a simultaneous increase of the quality of software;
4. *Continuous Execution:* the objective of this area is to continuously execute tasks that lead to a more stable, compliant, and better products;
5. *Continuous Model Improvement:* the purpose of this area is to continuously reflect on the assessment model and improve the interaction of assessment results and the suggested improvement for the software development process;
6. *Test Strategy:* the main purpose of this area is to provide an environment that allow the verification of the correct behavior and ensure the software quality for various software variants that are developed in a fast development pace within the software development;
7. *Communication:* the objective of the Communication Area is to verify the communication of all participating roles to avoid knowledge silos and to react on customer requirements faster.

Considering these areas, the review has led us to take the SAMI agile maturity model [14, 17] as a basis to develop our proposed model. Ozcan-Top and Demirörs [19] have confirmed that the SAMI model is comprehensive and well-organized structure, an argument confirmed also in [16]. By reviewing the agile practices offered by the SAMI model and evaluating their applicability, SAMI can be viewed as providing agile practices that the Agile Product Lines (APL) require at several levels. Therefore, we have adopted these agile practices as the basis for the targeted maturity model to address the "APL team level practices". In fact, we have adopted the 40 original SAMI agile practices. Then we have adapted and extended the SAMI model in accordance with the Agile Product Line principles and practices defined in the main sources of Agile Product Line Engineering such as [1], [20], and [21].

4.1 Development of AgiPL-AM

In addition to the 38 original SAMI agile practices, we defined 49 Agile Product Line practices that are incorporated in the final version of the AgiPL-AM model. Precisely, both the SAMI agile practices and the APL practices went through a review and refinement by using the phase two of our research approach. These refinements and changes were applied with respect to the original agile practices of the SAMI model.

Considering the areas presented above, it was identified that the SAMI model covers mainly the area 3 (i.e. Agile Software Development) and partially the area 2, area 4, area 5, and area 6. In our proposed model we have defined a new cate-gory called "Category 6 – Product line Architecture". Moreover, in our proposed assessment model we have conserved the five agile levels of SAMI model. Here-after we presented the different added practices. Due to lack of space, we have presented the new defined agile practices for each category separately.

Category 1 – Embrace change to deliver customer value. Table 1 presents the practices of each level that belong "Category 1". We have held back 5 practices from SAMI model and defined 5 new practices, namely L2.C1.2, L2.C1.3, L2.C1.4, L3.C1.1, L3.C1.2, and L4.C1.3.

For example, the description of the practice "L1.C1.1 – Reflect and tune process" is the following:

> *Holding retrospectives at regular intervals of the development process. The objective of this practice is to overcome process challenges that have been faced thus far [1].*

In addition, the practice "L2.C1.2 – Domain requirements" is an APL practice and the description of this practice is the following:

> *Domain requirements encompass requirements that are common to all applications of the software product line as well as variable requirements that enable the derivation of customized requirements for different applications [2].*

Table 1. Practices of "Category 1" of AgiPL-AM model

Levels	Practices
Level 1	L1.C1.1 – Reflect and tune process [14]
Level 2	L2.C1.1 – Evolutionary requirements [14, 17]
	L2.C1.2 – Domain Requirements [24]
	L2.C1.3 – Smaller, more frequent release [23]
	L2.C1.4 – Requirements discovery [23]
Level 3	L3.C1.1 – Regular reflection and adaptation [13]
	L3.C1.2 – Customer feedback is accessed for new features and ideas [6]
Level 4	L4.C1.1 – Client driven iterations [14]
	L4.C1.2 – Continuous customer satisfaction feedback [14]
	L4.C1.3 – Lean requirements at scale [23]
Level 5	L5.C1.1 – Low process ceremony [14]

Category 2 – Plan and deliver software frequently. Table 2 presents the practices of the "Category 2". In this category, 7 agile practices were held back from SAMI model and 10 APL practices were adopted. The adopted practices are L2.C2.2, L2.C2.3, L2.C2.4, L2.C2.5, L2.C2.6, L3.C2.3, L3.C2.4, L3.C2.5, L4.C2.3, and L5.C2.2.

Table 2. Practices of "Category 2" of AgiPL-AM model

Levels	Practices
Level 1	L1.C2.1 – Collaborative planning [14]
	L2.C2.1 – Continuous delivery [14, 17]
	L2.C2.2 – Two-tier planning and tracking (i.e. Domain Engineering tier & Application Engineering tier) [24]
Level 2	L2.C2.3 – Two-level planning and tracking [23]
	L2.C2.4 – Agile Estimating and Velocity [23]
	L2.C2.5 – Release planning [23]
	L2.C2.6 – Work product list [6]
	L3.C2.1 – Risk driven Iterations [14]
	L3.C2.2 – Plan features not tasks [14]
Level 3	L3.C2.3 – Mastering the iteration [23]
	L3.C2.4 – DoD after each software release [23]
	L3.C2.5 – Backlogs and Kanban Systems [23]
	L4.C2.1 – Smaller and more frequent releases [14]
Level 4	L4.C2.2 – Adaptive planning [14]
	L4.C2.3 – Measuring business performance (Project measure; Quality measure; Risk measure; Delivery record) [6, 23]
Level 5	L5.C2.1 – Agile project estimation [14]
	L5.C2.2 – Audit activities [6]

Category 3 – Human centricity. Table 3 presents category 3. We have adopted 5 agile practices from SAMI model and we have defined 3 new APL practices, namely, L2.C3.1, L4.C3.2, and L5.C3.2.

Table 3. Practices of "Category 3" of AgiPL-AM model

Levels	Practices
Level 1	L1.C3.1 – Empowered and motivated teams [14]
	L1.C3.2 – Collaborative Teams [14]
Level 2	L2.C3.1 – The Define/Build/Test teams of each tier: Domain Engineering tier & Application Engineering tier [23]
Level 3	L3.C3.1 – Self-organizing teams [14]
	L3.C3.2 – Frequent face-to-face communication [14]
Level 4	L4.C3.1 – Managing highly distributed teams [23]
Level 5	L5.C3.1 – Ideal agile physical setup [14]
	L5.C3.2 – Changing the organizations [16]

Category 4 – Technical excellence. This section is dedicated to the different practices of the “Category 4” introduced in the Table 4. In fact, the Category 4 of the AgiPL-AM has 24 practices among these practices 14 practices were held back from SAMI model. Moreover, 10 new APL practices have been defined, namely, L1.C4.1,

L1.C4.5, L1.C4.6, L2.C4.4, L2.C4.5, L3.C4.1, L3.C4.3, L3.C4.4, L4.C4.3, and L4.C4.4.

Table 4. Practices of "Category 4" of AgiPL-AM model

Levels	Practices
Level 1	L1.C4.1 – Product Backlog [16, 23]
	L1.C4.2 – Coding standards [16]
	L1.C4.3 – Knowledge sharing [14]
	L1.C4.4 – Task volunteering [14]
	L1.C4.5 – Continuous and automated tasks [6]
	L1.C4.6 – Acceptance testing [16]
Level 2	L2.C4.1 – Software configuration management [14]
	L2.C4.2 – Tracking iteration progress [14]
	L2.C4.3 – No big design up front [14]
	L2.C4.4 – Automated testing [6]
	L2.C4.5 – Bidirectional traceability record [6]
Level 3	L3.C4.1 – Continuous deployment [6]
	L3.C4.2 – Continuous integrating [14]
	L3.C4.3 – Scalable and Continuous tests [6]
	L3.C4.4 – Continuous compliance [6]
	L3.C4.5 – Continuous improvement (refactoring) [14]
	L3.C4.6 – 30% of "level 2" and "level 3" people [14]
	L3.C4.7 – Unit tests [14]
Level 4	L4.C4.1 – Daily progress tracking meetings [14]
	L4.C4.2 – User stories [14, 23]
	L4.C4.3 – Adaptive test strategy [6]
	L4.C4.4 – Improvement opportunity and plan [6]
Level 5	L5.C4.1 – Test driven development [1]
	L5.C4.2 – no or minimal number of Cockburn Level "1B" or "-1" [14]

Category 5 – Customer collaboration. This category has 15 agile practices. According to the proposed assessment model, 8 APL practices were defined and 7 practices were adopted from SAMI model. Table 5 presents the practices of the "Category 5" of AgiPL-AM. Precisely, L1.C5.2, L1.C5.3, L1.C5.4, L3.C5.1, L3.C5.2, L4.C5.3, L4.C5.4, L4.C5.5, and L5.C5.2.

Category 6 – Product-Line Architecture. This category is a new category added to the 5 five categories of SAMI model. In this category, the product line principles related to Agile Product Lines architecture are gathered. Here, 13 APL practices were defined. These new practices of the "Category 6" of AgiPL-AM model are presented in Table 6.

Table 5. Practices of "Category 5" of AgiPL-AM model

Levels	Practices
Level 1	L1.C5.1 – Customer commitment to work with development team [6, 14]
	L1.C5.2 – Destructed "Knowledge silos" [6]
	L1.C5.3 – Fast feedback channels [6]
	L1.C5.4 – Common understanding for the SPL [6]
Level 2	L2.C5.1 – Customer contract reflective of evolutionary development [14]
Level 3	L3.C5.1 – Direct communication channels [6]
	L3.C5.2 – Vertical commitment [6, 24]
Level 4	L4.C5.1 – "CRACK" Customer immediately accessible [14]
	L4.C5.2 – Customer contact revolve around commitment of collaboration [14]
	L4.C5.3 – DevOps (Integrated Development and Operations) [16]
	L4.C5.4 – Vision, features [4]
	L4.C5.5 – Impact on customers and operations [16, 23]
Level 5	L5.C5.1 – Frequent Face-to-face interaction between develops and users [14]
	L5.C5.2 – Concurrent testing [16]

Table 6. Practices of "Category 6" of AgiPL-AM model

Levels	Practices
Level 1	L1.C6.1 – Software Product Line architecture [6]
	L1.C6.2 – Reuse strategy [24]
	L1.C6.3 – Modular software architecture [6]
Level 2	L2.C6.1 – Distributed development [6, 23]
	L2.C6.2 – Modularity of software components [6]
	L2.C6.3 – Reusable software units [6]
Level 3	L3.C6.1 – SPL architecture is open to changes and refactoring is possible [6]
	L3.C6.2 – Collaboration with supplier is improved [6]
Level 4	L4.C6.1 – Fast changes in requirements [6, 24]
	L4.C6.2 – Adequate scoping process [24]
	L4.C6.3 – Intentional architecture [23]
Level 5	L5.C6.1 – Standardized interfaces for software units [6]
	L5.C6.2 – Continuously evaluation of the architecture [6]

4.2 The AgiPL-AM

Based on the iteration development and the retrospective of followed approach, several adjustments were done in order to optimize the AgiPL-AM model, which involves both agile practices adopted from the SAMI model and APL practices that were defined to address the main objective. The main changes that are performed on the agile practices adopted from the SAMI model are the following:

- The agile practices "Paired programming" and "Agile documentation" were removed as their purposes are covered by "L2.C3.1 – Define/Build/Test teams of

each tier: Domain Engineering tier & Application Engineering tier" and "L2.C1.2 – Domain Requirements";
- The agile practices "Backlog" was renamed into "Product Backlog" in order to match more the actual agile terminology and thus provide a better representation;
- The practice "Product Backlog" was moved to the "level 1 – Collaborative" since it is considered to provide the basis for other practices at higher maturity levels.

When the agile/APL practices were defined and refined, a set of governing rules were applied in order to populate these practices in the appropriate maturity level and principle. These rules are the followings:

- The *first rule* states that each practice has to contribute to the achievement of the maturity level objective in which it is positioned. For example, the practice "L1.C3.2 – Collaborative Teams" should addresses directly the "collaboration" objective of maturity Level 1 (i.e. Collaborative);
- The *second rule* is followed to ensure the relevancy of the practice with respect to the agile principle that it is associated. The practice L1.C3.2 is related to the principle for "Human-centricity";
- The *third rule* states that the relation between the practices in such a way that practices positioned at higher levels depend on the achievements of the practices at lower levels. For example, the APL practice of "L2.C2.2 – Two-tier planning and tracking (i.e. Domain Engineering tier & Application Engineering tier)" at level 2 depends on achieving some of "Level 1" practices, such as "L1.C3.2 – Collaborative Teams" and "L1.C2.1 – Collaborative planning".

The final version of the AgiPL-AM model was optimized and its adopted practices are presented above in *section 4.2*. AgiPL-AM is considered as a descriptive model (i.e. as opposed to prescriptive) since it describes only the essential practices that an organization that adopt an APL approach should possess at a particular level of maturity.

In our proposed model, on the one hand, the agile practices adopted from SAMI model are assessed by using the original indicators of SAMI model. On the other hand, the APL practices are assessed by using AgiPL-AM indicators (i.e. as set of defined indicators related to the APL practices defined as part of AgiPL-AM). These indicators are not listed in this paper due to the lack of space. For example, in order to assess the practice "L3.C2.3 – Mastering the iteration", the following indicator is used: ***"L3.C2.3.ind** – the development team has effective iterations consisting of sprint planning, tracking, execution, and retrospectives".*

Furthermore, based on the practices of AgiPL-AM, all the indicators are rated by using an achievement scale. From the ISO/IEC 15504 assessment standard [27], the rating scheme was adopted. This rating scheme is the following:

i. **(N) – "Not achieved (0% −35%)"** represents little or no evidence of achievement of the practice;

ii. **(P) – "Partially achieved (35% − 65%)"** denotes some evidence of an approach to, and some achievement of the practice. Some aspects of achievement may be unpredictable;
iii. **(L) – "Largely achieved (65% − 85%)"** indicates that there is evidence of a systematic approach to, and significant achievement of the practice; despite some weaknesses;
iv. **(F) – "Fully achieved (85% – 100%)"** indicates strong evidence of a complete and systematic approach to, and full achievement of the practice without any significant weaknesses.

The assessment process requires going through all practices and corresponding indicators to assess the entire set of practices in AgiPL-AM. In order to provide confirmation regarding the results of the assessment, an assessment report should compiled to present the results to relevant parties.

5 Application of AgiPL-AM

In this section, we apply the AgiPL-AM model to assess the adoption level of agile development within an Agile Software Product Line approach, namely, the ScrumPL approach [8].

According to Santos and Lucena [8], the ScrumPL process is intended to de-velop agile software product lines (APLs) by combining engineering activities from Software Product Line Engineering with the Scrum method. ScrumPL is composed on the one hand, by the Scrum lifecycle phases, namely, Planning, Staging, Development, and Release. On the other hand, by the Software Product Line Engineering (SPLE) stages, that is, Domain Engineering and Application Engineering. The Scrum phases and the SPLE sub-processes are combined to form ScrumPL.

By repeating the rules applied in developing the proposed model, AgiPL-AM has been developed in such a way that each practice contributes to the founda-tion required for the practices that are at higher maturity levels. For example, the agile practice of level 2 "L2.C2.5 – Release planning" provides necessary basis for the practice of level 4, which is "L4.C2.2 – Adaptive planning". Thus, focus-ing the attention on 'Level 3' practices without satisfying the 'Level 2' ones will be ineffective. Therefore, it is expected during the assessment process to have more practices satisfied at lower levels than at higher levels.

Figure 2 summarizes the results of assessing the approach ScrumPL by applying our proposed model AgiPL-AM. It is clear that the level of achievement tends to decrease towards higher maturity levels. However, the practices that are "Not Achieved" are spread over all levels. ScrumPL achieves only "6.9 %" of the practices. "28.7 %" of the practices are not achieved at all. Moreover, 33.4 % (i.e. 29 practices) of the practices are largely achieved whereas, 31 % of the practices are partially achieved.

Level 1 represents the collaborative level and has 17 practices. Among these practices one practice is 'not achieved', six practices are 'partially achieved', and three practices are APL practices. Thus, just seven practices are 'largely achieved' and 'Fully achieved'.

Fig. 2. Results of the Assessment of ScrumPL using AgiPL-AM model

Accordingly, the collaboration issue is not strongly ensured by ScrumPL approach. At level 3, which represents the effectiveness level, only five practices (i.e. 5 out of 20) are 'largely achieved' the other practices either 'partially achieved' or 'not achieved' at all. This means that the ScrumPL approach lacks practices that ensure its effectiveness.

By using the AgiPL-AM approach, the strengths and the weakness of the ScrumPL method were identified. In fact, the model has highlighted all the agile and APL practices that are not covered by ScrumPL. Thus, the results of the assessment could be used to improve ScrumPL or even to define a new APL approach. For example, at "Level 3" three practices are not achieved. These are the "L3.C1.2 – Customer feedback is accessed for new features and ideas", "L3.C3.2 – Frequent face-to-face communication", and "L3.C5.1 – Direct communication channels". At this level, a special situation is manifested as a communication barrier between the user representatives and the development team members, which prevented them to establish a close integration of development and operations. These subjects of weaknesses in the lower maturity levels were indicated as the most prominent points on which any company should direct its attention when adopting ScrumPL.

6 Conclusion

The combination of agile software development and software product lines is a promising approach. The current status on the agile adoption within agile soft-ware product line approaches is hard to define [6], thus, it was identified the need for a specific assessment model for assessing the situation of agile adoption with-in agile product line approaches.

The research objective of this paper is to design an assessment model that can be used as a guideline by organizations to adopt agile product line methodologies and assess the success level of agile practices adoption. Through the review of the literature it was identified that known of the studied assessment models focus simultaneity on agile and APL practices in detail within APL approaches. Com-paring to these approaches, AgiPL is considered as a structured approach that increases the chances of success in agile and APL practices within agile software product lines. In addition, AgiPL serves as an evolutionary path that increases organization's agile maturity. Also, the proposed model prioritizes the improve-ment actions in adopting agile and APL practices. The illustrated example in this paper shows the applicability of the assessment model.

The proposed work is an ongoing work. For future work, we plan to further evaluate the AgiPL-AM model in order to improve AgiPL-AM model. As next step, we will validate AgiPL-AM empirically and we will involve a number of members of companies in the evaluation of the applicability of AgiPL-AM, in assessing the level of agility of their companies, and in evaluating the overall findings of the assessment model.

References

1. da Silva, I. F., Neto, P., O'Leary, P., de Almeida, E., and de Lemos Meira, S.R.: Agile Software product lines: a systematic mapping study. Soft. Prac. Exp., 41(8), 899–920.
2. Tian, K. and Cooper, K.: Agile and Software Product Line Methods: Are They So Different? In: 1st international workshop on agile software product line engineering, pp. 1-8.
3. Carbon, R., Lindvall, M., Muthig, D., Costa, P.: Integrating Product Line Engineering and Agile Methods: Flexible Design Up-front vs. Incremental Design. In: 1st International Workshop on Agile Product Line Engineering (2006).
4. Boehm, B.W.: Get Ready for Agile Methods, with Care. IEEE, 35(1), pp. 64-69 (2002).
5. Navarrete, F., Botella, P., Franch, X. : How Agile COTS Selection Methods are (and can be). In: Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications. pp. 160-167 (2005).
6. Hohl, P., Münch, J., Schneider, K., and Stupperich, M.: Real-Life Challenges on Agile Software Product Lines in Automotive. In: PROFES'17, pp. 28-36 (2017).
7. Klünder, J., Hohl, P., and Schneider, K.: Becoming Agile while preserving software product lines: an agile transformation model for large companies. (ICSSP '18), pp. 1-10.
8. dos Santos Jr., A. F., Lucena Jr, V. F. : SCRUMPL - Software Product Line Engineering with Scrum. In: Proceedings of ENASE 2010, pp. 239-244 (2010).
9. CMMI Product Team.: CMMI for development, Version 1.3: Improving processes for developing better products and services. Technical Report. (2010).
10. Jasmine, K. S., and Vasantha, R.: A new capability maturity model for reuse based software development process. Int. J. of Eng. and Technology, 2(1), pp. 112-116 (2010).
11. VDA QMC Working Group 13 / Automotive SIG. Automotive spice process assessment / reference model. (2017).
12. Qumer A, Henderson-Sellers B.: A framework to support the evaluation, adoption and improvement of agile methods in practice. J. of Sys. and Soft., 81(11), pp. 1899–1919 (2008).
13. Schweigert T., Vohwinkel D., Korsaa M., Nevalainen R., Biro M.: Agile Maturity Model: A Synopsis as a First Step to Synthesis. In: vol 364. Springer, pp. 214-227 (2013).
14. Sidky, A., Arthur, J., and Bohner, S. A disciplined approach to adopting agile practices: the agile adoption framework. Inno. in Sys. and Soft. Eng., 3(3), pp. 203-216 (2007).
15. ManifestoAgile: Manifesto for Agile Software Development (2001).
16. Turetken, O., Stojanov, I., and Trienekens, J. J. M.: Assessing the adoption level of scaled agile development: a maturity model for Scaled Agile Framework. *JSEP*, 29(6), (2017).
17. Sidky, A. and Arthur, J.: Agile adoption process framework - indicators document (2006).
18. Hevner A. R., March, S., Park, J., and Ram, S.: Design science in information systems research. MIS Quarterly 2004, 28(1), pp. 75-105 (2004).
19. Ozcan-Top O., Demirörs O.: Assessment of Agile Maturity Models: A Multiple Case Study, vol 349. Springer, Berlin, Heidelberg, pp. 130–141 (2013).
20. Díaz, J., Pérez, J., Alarcón, P. P., and Garbajosa, J.: Agile product line engineering—A systematic literature review. Soft. Practice and Experience, 41(8), pp. 921–941 (2011).
21. Farahani, F. F., and Ramsin, R.: Methodologies for Agile Product Line Engineering: A Survey and Evaluation. In: SoMeT_14, Amsterdam: IOS Press BV, pp. 545-564 (2014).
22. Hohl, P., Stupperich, M., Münch, J. and Schneider, K.: An Assessment Model to Foster the Adoption of Agile Software Product Lines in the Automotive Domain. In (ICE/ITMC), 1-9
23. Leffingwell, D. Agile software requirements: lean requirements practices for teams, programs, and the enterprise, Addison-Wesley Professional (2011).
24. Apel, S., Batory, D., Kästner, C., and Saake, G. Feature-oriented Software Product Lines: Concepts and Implementation. Berlin: Springer-Verlag (2013).