Towards an Efficient Traceability in Agile Software Product Lines

Zineb Mcharfi, Bouchra El Asri, Ikram Dehmouch
IMS Team, SIME Laboratory
ENSIAS, Mohammed V Rabat University
Rabat, Morocco
{zineb.mcharfi@gmail.com, elasri@ensias.ma, ikram.dehmouch@gmail.com}

***Abstract*—In a volatile market, where it is difficult to predict future needs, classical Software Product Lines show limitations and become pricey. Therefore, researchers managed to add supplements in order to reach flexibility, and this led to the Agile Product Line Engineering concept. However, this concept has not gained yet sufficient maturity, and works are still necessary to establish the best practices for putting Agile Product Line Engineering into practice, especially when it comes to their traceability. In this paper, we discuss the correlation between agility and traceability dimensions through the state of the art of traceability in Agile Software Product Lines, and present our solution based on markers and break-even point in order to establish a traceability methodology in Agile Software Product Lines.**

Keywords-Software Product Lines; Agile Software Product Lines; traceability; efficient traceability.

I. INTRODUCTION

Considering market growth and competitiveness, companies try to achieve mass customization with lower costs, reduce time to market, and insure product quality while getting customer's satisfaction. From a software engineering point of view, Software Product Lines (SPL) is a promising concept that helps dealing with those challenges [1][2].

However, in some business environments, SPL may not be enough reactive compared to market growth. In fact, designing a SPL requires deploying important efforts and time in order to speculate on future products and functionalities that may be needed. Also, the Return On Investment (ROI) of those efforts might be very small in a volatile market [3]. Those constraints pushed developers and researchers to look for improving SPL in order to gain flexibility, which led to the concept of Agile Product Line Engineering (APLE) [4]–[6].

Many researchers worked on the feasibility of combining SPL and Agile Software Development (ASD) [3]–[6], as both of them share the same objectives of increasing productivity and software quality while optimizing production time, even if they present differences in the concept and practices [4]. Traceability might be considered as one of the challenging points in combining SPL and agility; the former, because of its complexity and need to manage variability, requires traceability documentation to assure consistency of the links between artifacts and facilitate changes implementation [2], while the latter advocates less use of documents [7].

In the present paper, we will illustrate, throughout a state of the art, how the existing works manage traceability in their Agile Software Product Lines (ASPL), depending on the agile method used. We will also present our contribution, a methodology based on the concepts of "markers" and "break-even point" for an efficient traceability in ASPL.

The remainder of this paper is structured as follow: in Sections II, we describe the concepts of SPL, ASD and ASPL. Section III presents the traceability in SPL and a state of the art of traceability in ASPL. We discuss our contribution in Section IV and illustrate it in a case study in Section V, before concluding in Section VI.

II. BACKGROUND AND MOTIVATIONS

In this section, we will first briefly introduce SPL and ASD in order to present later the ASPL, a concept based on the combination of the two previous ones.

A. Software Product Lines

As defined by Northrop [1], a SPL is *"a set of software-intensive systems that share a common, managed feature set satisfying a particular market segment's specific needs or mission and that are developed from a common set of core assets in a prescribed way"*. It is used in the organizations that produce numerous products answering specific needs, but having many components in common. Those common components (e.g., architecture, requirements, test plans, schedules, budgets and processes description) are called "core assets". Adopting a SPL approach allows to produce new systems by reusing the existing ones, in an organized manner.

Accordingly, SPL is a combination of three major interacting elements, called the SPL essential activities [1][8]: (1) core asset development or Domain Engineering (DE), (2) product development or Activities Engineering (AE) and (3) technical and organizational management that orchestrates those two activities.

 ISBN: 978-1-61208-367-4

SPL is by far considered as an up-front, proactive (in opposite to reactive) reuse demarche [9]: it is based on a production plan, involves both technical and organizational management, is a direct consequence of the organization strategy, and it is used to reach predictable results.

B. Agile Software Development

ASD is a concept based on the Agile Manifesto [7]. As for SPL, ASD seeks to satisfy customer needs rapidly, while insuring a good software quality, yet unlike SPL, the ASD concept is based on simplicity, iterations, and reducing up-front design [5].

ASD values, described in the Agile Manifesto, are "individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation and responding to change over following a plan" [7].

The Agile Manifesto defines also twelve principles for ASD [7]. Hereinafter some: (1) customer satisfaction by rapid delivery of useful software, (2) welcome changing requirements, even late in development and (3) regular adaptation to changing circumstances.

Thus, ASD shows values where SPL shows weaknesses, especially when it comes to flexibility and adaptation to changing requirements and circumstances.

Accordingly, some complementarity can be found between SPL and ASD, which led to the APLE concept.

C. Agile Product Line Engineering: Software Product Lines combined to Agile Software Development

As explained earlier, SPL need an up-front design, with heavy processes and significant efforts. It helps answering planned changes, but if it comes to unstable environments with rapidly changing conditions, the investment in SPL might be pricey [3]. On the other hand, ASD seeks to satisfy customer requirements in a reactive way, promoting continuous discussion with the customer, and avoiding up-front developments.

According to Díaz et al. [3] and Ghaman et al. [5], the combination of SPL and ASD principles allows eliminating long term investment in up-front design, especially in volatile markets where it would represent a non-profitable investment in the long term with huge losses due to no-longer useful core assets or never used ones. It allows also dealing with situations where there is lack of knowledge about domain engineering, or where no speculation can be made.

Many works discuss the application of agility to SPL: In [6], agility is used in the design phase and the benefices of its introduction by gaining in speed are demonstrated. Noor et al. [10] used a collaborative approach to introduce agility when planning and scooping the Product Line (PL). They used some agile development principles, such as valuing customer collaboration and high degree of flexibility. Urli et al. [11] described the application of agility for SPL evolution through a case study, using Composing Feature Models (CFM); they first built an information broadcasting system for a limited academic structure, but then had to deal with larger institutions and numerous customers, which represented multiple devices and sources of information. Therefore, they used a SPL demarche for the re-engineering of their system and, as they had to interact continuously with the customers, they sought lightness and introduced agility to their approach. This decision helped them reach simplicity (they decomposed the requirements in features with fine granularity) and be more reactive to the customers' needs. Another approach was established by Ghanam and Maurer [12], who used a Test Driven Development (TDD) method to deal with agility in SPL. They introduced SPL demarche in an agile environment that uses eXtreme Programing (XP), and instead of using requirement documents to begin development, they used Acceptance Tests (AT) generated through the XP process as test artifacts, which are the basis for the model adopted.

III. Traceability in Agile Software Product Lines

In such a complex environment (i.e., ASPL), where we have to manage variability in a constantly evolving context, it is very important to insure traceability along the software development process.

However, based on the observation made by the review in [3], and completed with our literature analysis, we noticed that very few researches deal explicitly with the problematic of traceability in ASPL, knowing that managing traceability is very important in such evolving environments. Therefore, we choose to discuss the problematic of traceability in ASPL, given the challenges that it presents.

A. Traceability in Software Product Lines

Traceability helps follow the components' life, link between different software artifacts, from requirements to source codes and backwards and, in a larger scale, helps verify that all requirements have been implemented and the artifacts documented [13]. It is also a mean to consider different architecture choices and identify errors, and to facilitate communication between stakeholders [14]. Traceability is very helpful when it comes to maintenance and evolution as it allows analyzing and controlling the impact of changes [15].

SPL add complexity to the traceability due to their reuse characteristics and the variability management [16]. Berg et al. [17] proposes a conceptual variability model to deal with traceability in SPL and consider that, in addition to the two dimensions of traceability in a simple software (i.e., phases of development and levels of abstraction), for SPL there is need to add variability as a third dimension. They propose to handle SPL variability, and especially the traceability problematic, by adopting a three dimensions conceptual variability model that uses feature modeling to manage variability and traceability. Anquetil et al. [14][16] added a fourth dimension, namely evolution, to link between the different versions of every artifact, and a fifth one, versioning, to trace components' changes in time.

In the next section, we will draw up a state of the art of traceability in ASPL, based on the five traceability dimensions, as presented by Anquetil et al. [14]: (1) refinement traceability that links abstract artifacts to more concrete ones that realize them (no variability), (2) similarity

 ISBN: 978-1-61208-367-4

traceability for links between artifacts at the same level of abstraction (requirements, design, etc.), (3) use-variability traceability for instantiation links (from DE to AE), (4) realize-variability traceability to link between the variant and the artifact that realizes it at the DE level, (5) versioning traceability to link two successive versions of an artifact.

B. State of the art of traceability in Agile Software Product Lines

In this section, we will draw up a listing of works that present a methodology for introducing agility in SPL, and discuss those methods according to the following questions: (1) What are the traceability dimensions (according to [14]) does the presented methodology cover? (2) Which agile method is used? (3) At which stage of SPL is the agility introduced? (4) How does it deal with traceability?

Our first observation is that not all the methodologies found in literature propose solutions that take into consideration traceability (Table I). One assumption might be that it depends on the stage where agility is used. In fact, in [24], agility was applied in scoping, and all the work was focalized on it. In the other papers, at least refinement traceability is covered.

Papers that approach the architectural problematic and variation points [12][18][22][23] cover another dimension: variability traceability, with a link type "realize". The approach presented in [12][18], which uses acceptant tests, and the one that combines workflow and web services [22] handle also the variability traceability with a link type "use".

TABLE I. WORKS ANALYSIS ACCORDING TO TRACEABILITY DIMENSIONS

Traceability dimensions				
Refinement	***Similarity***	***Variability (Use)***	***Variability (Realize)***	***Versioning***
[10][12][18]–[22]	-	[12][18][22]	[12][18][22][23]	[19]

TABLE II. WORKS ANALYSIS ACCORDING TO THE AGILE METHOD

Reference	***Agile method***	***Level of agility application***	***How traceability is applied***
[23]	Scrum	Architecture	Using Product Line Architectural Knowledge (PLAK) metamodel and Design decision by documenting adding features and changing features
[12][18]	XP	Requirements	AT
[24]	Agility principles	Scoping	-
[10]	Agility principles applied though Collaboration Engineering	planning	ThinkLets + collaborative process
[19]	Evo	Requirements management	Impact Estimation Tables (IET)
[20][21]	Agility principles and XP at « Preparing for Derivation » phase	Product Derivation	-
[26]	Some agile principles (Flexible, quick, adaptable, user-oriented)	Design to architecture	WebServices + workflow + WebPads-based approach

In [19], versioning traceability is addressed through the use of Impact Estimation Tables. Iterations (and accordingly components changes) are listed for each goal per project and per release.

In general, there is a lack in covering several traceability dimensions in ASPL approaches literature. Also, concerning the agile methods (Table II), many works are based only on agility principles [10][20][24]. XP approach is also widely used [12][18][20][21]. However, by using AT, [12] and [18] cover three of the five traceability dimensions and propose an approach that covers the entire process, from requirements to code units.

IV. OUTLOOK AND CONTRIBUTION

Based on our researches, we found the study of ASPL a challenging field that did not gain yet sufficient maturity, especially when it comes to managing traceability. In fact, in case of ASPL, we need to consider the agile characteristics of the environment. Adding agility means frequent requirements' change, even late in development, and continuous interaction with customers. Also, while agility tries to avoid heavy processes and excessive documentation, traceability needs more produced and maintained documents.

In the works related to ASPL, as discussed in the previous section, there is lack of managing traceability: not all the ASPL methodologies proposed in literature deal with traceability and, for those taking it into consideration, the agile configuration proposed doesn't allow tracing the whole PL chain, according to the five traceability dimensions detailed in [14].

Moreover, referring to our literature analysis, we noted that only papers presenting an automated refactoring approach used traceability in an efficient way: only really affected elements in the SPL are localized and modified before rebuilding the SPL [12][18][19].

Thus, for our contribution, we propose a methodology based on markers for efficient traceability in an ASPL environment: in a SPL, every produced element is the result of specific concatenation and instantiation of some product line components. Knowing this combination helps tracing efficiently the product generation path by targeting only the concerned components. Based on this observation, we are establishing an approach that consists of adding a marker to every SPL component. Each marker is unique and encapsulates the component characteristics and, as a product is the result of specific core assets instantiation, it inherits from those core assets' characteristics. Therefore, the idea is to identify the product with a marker composed from the corresponding core assets' ones, and to create a link between the components and the products, based on those markers. The marking step is added to the core assets generation

 ISBN: 978-1-61208-367-4

process (see Figure 1). However, a special treatment is reserved to variation points. In fact, variants share characteristics with their parent components (i.e., variation points). Thus, instead of generating a new marker for the variants, we use a function for "marker mutation". This function allows modifying the variation point marker to generate the variant's one, while keeping the former's characteristics and adding the latter's specific ones. It helps lighten the process, as we are in an agile environment, and establishing a realize-variability traceability link. By the end of this process, a multidimensional marking matrix is generated. Its dimensions correspond to core assets and its cells to a combination of the corresponding markers. Thus, each derived product is distinguished by a marker that corresponds to a specific cell in the multidimensional matrix (see Figure 2). However, in order not to complicate the matrix and to preserve the agility of the environment, (tracing the whole product generation process might be heavy and costly, and even useless regarding the traceability purpose in the developed ASPL), we introduce the concept of "break-even point". It represents the point of balance between the desired level of traceability detail and the costs of building and maintaining the system. It is flexible and depends on the level of traceability needed. The aim of this break-even point concept is to define a traceability limit based on which we select only the core assets needed for the product traceability. The product marker is then assembled depending on the composition of those core assets in the product. In order to define the parameters to consider for establishing a break-even point, we are conducting a study to outline the limitations of traceability in an agile environment. We aim to determine, through this study, the level of traceability that does not penalize the agility of the ASPL, and we intend to evaluate this approach using graph theory principles, to prove that the selected subgraph (connection of core assets) can effectively allow tracing the products' generation paths, knowing the environment constraints.

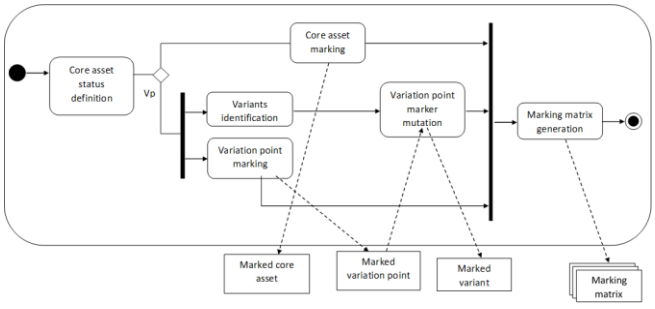

Figure 1. Core assets marking process

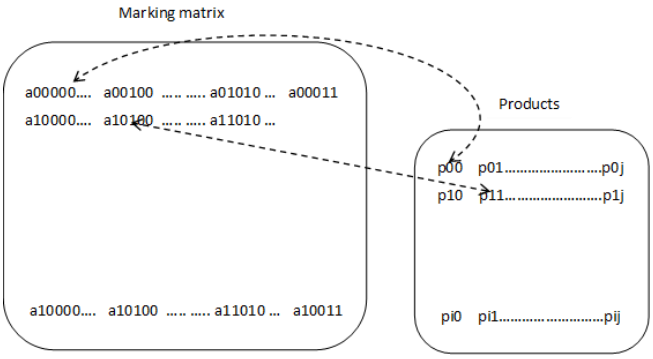

Figure 2. Link between products and marking matrix

V. CASE STUDY

To illustrate our approach, we present hereafter a case study of offers implementation in the case of a telecommunication operator.

Telecommunication market is very competitive and each operator has to be reactive to the market changes. Also, with the expansion of smartphones and intelligent home equipment, trend is for broadband, high speed data transmission, and free short messages and calls. Therefore, offers share the same objectives but present them in different ways, depending on the proposed services, the pricing and the customer's subscription. Moreover, to reach reactivity, the telecommunication operator needs to propose new offers with new services frequently. Considering those elements, and in order to optimize development and deployment costs, providers of network solutions use ASPL to implement the offers: stakeholders (i.e., marketing staff) are continuously involved and offers frequently changing (agility); they share the same bases (common components) and differ depending on the services proposed and the customer's subscription (variation points).

Another telecommunication market constraint concerns revenue problematic: a critical error generated after deploying an offer may cause important financial losses if not quickly fixed, depending on the volume of traffic and data transmission. That's why reactivity in tracing product generation path is very important.

With our approach, each generated product will have a marker composed from those of its components. Thus, we can easily identify the concerned elements to be checked and fixed. We can also identify the other impacted products and the related test cases to execute them and verify the product integrity (see Figure 3). When an offer is initiated by the management (based on market statistics and indicators, decision making system, etc.), it is implemented as a result of the instantiation of concerned components (use cases, design components, realization components and test scenarios) of the ASPL. Each component has its unique marker (UCi, DCi, RCi and Ti) and the generated product's marker (UC1, DC1, RC1, RC2, T1, T2, T3) is a result of their concatenation.

 ISBN: 978-1-61208-367-4

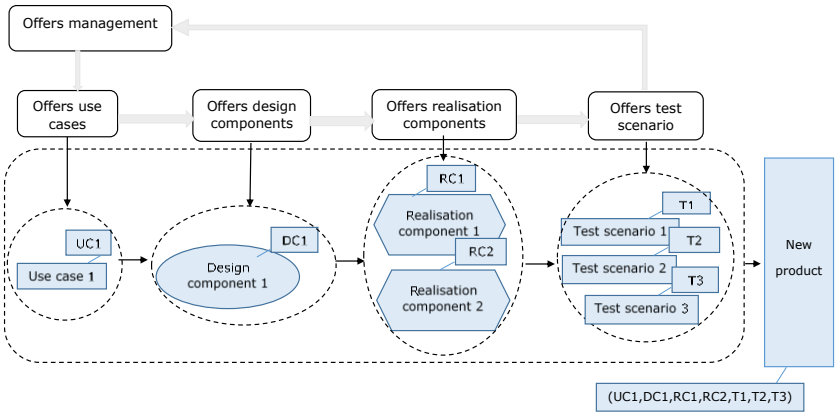

Figure 3. Simplified case study for telecommunication offers implementation

VI. CONCLUSION AND FUTURE WORK

Agile Product Line Engineering is a new promising method in software engineering. It helps companies gain flexibility, reactivity and customer satisfaction in a volatile and competitive context while optimizing costs and efforts.

We discussed in this paper the problematic of traceability in an ASPL through a state of the art, and proposed an approach for ASPL traceability based on markers and break-even points.

As the implementation of a break-even point requires a balance between the desired level of traceability and the costs of building and maintaining the agile system, our future contribution will focus on the optimization of the granularity and depth level of traceability in an ASPL.

REFERENCES

[1] L. M. Northrop, "SEI's software product line tenets, "IEEE Softw., vol. 19, no. 4, 2002, pp. 32–40.

[2] K. Pohl, G. Böckle, and F. Van Der Linden, Software product line engineering. 2005.

[3] J. Díaz, J. Pérez, P. P. Alarcón, and J. Garbajosa, "Agile Product Line Engineering - A Systematic Literature Review," Softw. Pract. Exp., vol. 41, no. 8, 2011, pp. 921–941.

[4] K. Tian and K. Cooper, "Agile and software product line methods: are they so different," 1st Int. Work. Agil. Prod. Line Eng. (APLE), collocated with 10th Int. Softw. Prod. Line Conf., 2006.

[5] Y. Ghanam, F. Maurer, and K. Cooper, "A Report on the XP Workshop on Agile Product Line Engineering," ACM SIGSOFT Softw. Eng. Notes, vol. 34, no. 5, 2009, pp. 25–27.

[6] R. Carbon, M. Lindvall, D. Muthig, and P. Costa, "Integrating Product Line Engineering and Agile Methods : Flexible Design Up-Front vs . Incremental Design," in 1st International Workshop on Agile Product Line Engineering APLE06, 2006, pp. 1–8.

[7] M. Fowler and J. Highsmith, "The agile manifesto," Softw. Dev., vol. 9, 2001, pp. 28–35.

[8] L. Northrop and P. Clements, "Software Product Lines," Carnegie Eng. Inst., 2005, pp. 1–105.

[9] C. Krueger, "Eliminating the adoption barrier," IEEE Softw., vol. 19, no. 4, Jul. 2002, pp. 29–31.

[10] M. a. Noor, R. Rabiser, and P. Grünbacher, "Agile product line planning: A collaborative approach and a case study," J. Syst. Softw., vol. 81, no. 6, Jun. 2008, pp. 868–882.

[11] S. Urli, M. Blay-Fornarino, P. Collet, and S. Mosser, "Using composite feature models to support agile software product line evolution," in Proceedings of the 6th International Workshop on Models and Evolution - ME '12, 2012, pp. 21–26.

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

[13] J. Cleland-Huang, O. Gotel, J. H. Hayes, P. Mäder, and A. Zisman, "Software Traceability: Trends and Future Directions," in ACM, 2014.

[14] N. Anquetil et al., "A model-driven traceability framework for software product lines," Softw. Syst. Model., vol. 9, 2010, pp. 427–451.

[15] Y. C. Cavalcanti et al., "Towards metamodel support for variability and traceability in software product lines," in Proceedings of the 5th Workshop on Variability Modeling of Software-Intensive Systems - VaMoS '11, 2011, pp. 49–57.

[16] N. Anquetil et al., "Traceability for Model Driven , Software Product Line Engineering 2 Software Product Line," ECMDA Traceability Work. Proc., 2008.

[17] K. Berg, J. Bishop, and D. Muthig, "Tracing software product line variability: from problem to solution space," Proc. 2005 Annu. Res. Conf. South Africain Inst. Comput. Sci. Inf. Technol. IT Res. Dev. Ctries., 2005, pp. 182–191.

[18] Y. Ghanam and F. Maurer, "An Iterative Model for Agile Product Line Engineering.," SPLC, 2008, pp. 377–384.

[19] B. G. K. Hanssen and T. E. Fægri, "Process fusion : an industrial case study on agile software product line engineering," J. Syst. Softw., 2008, pp. 843–854.

[20] P. O'Leary, F. M. Caffery, I. Richardson, and S. Thiel, "Towards agile product derivation in software product line engineering," 2009.

[21] P. O'Leary and F. McCaffery, "An agile process model for product derivation in software product line engineering," J. Softw. Evol. Process, 2012.

[22] M. Karam, S. Dascalu, and H. Safa, "A product-line architecture for web service-based visual composition of web applications," J. Syst. Softw., vol. 81, no. 6, 2008, pp. 855–867.

[23] J. Díaz, J. Pérez, and J. Garbajosa, "Agile product-line architecting in practice: A case study in smart grids," Inf. Softw. Technol., vol. 56, no. 7, Feb. 2014, pp. 727–748.

[24] M. Balbino, E. S. De Almeida, and S. Meira, "An Agile Scoping Process for Software Product Lines," in SEKE 2011 - Proceedings of the 23rd International Conference on Software Engineering and Knowledge Engineering, 2011, pp. 717–722.

 ISBN: 978-1-61208-367-4