Software Product Line Agility

Ahmed Abouzekry
Computer Science Department
Arab Academy for Science and Technology
Cairo, Egypt
abouzekry@yahoo.com

Riham Hassan
Computer Science Department
Arab Academy for Science and Technology
Cairo, Egypt
riham@cairo.aast.edu

***Abstract*— Software reuse constitutes a significant challenge for different development communities, while systematic reuse is a difficult target to achieve. Software Product Line (SPL) has been nominated as one of the effective approaches promoting software reuse. In this paper, we propose the Enterprise Product Line Software Process (EPLSP) that integrates practices of both the Enterprise Unified Process (EUP) and the Agile Unified Process (AUP). This integration benefits the engineering process with both reusable components architecture and fast time to market final products. EPLSP strategy focuses on the two major aspects of SPL namely the Core Assets (CA) and the Product Development (PD). CAs are those reusable artifacts and resources that form the basis for the SPL. PD involves building, acquisition, purchasing, retrofitting earlier work of software products, or any combination of these options. EPLSP promotes a clear up-front architecture in the CA while employing agility for PD. Constructing an up-front architecture for CA is effective in enhancing reusability and increasing productivity. Using agility in PD is meant to improve the time to market variable. We demonstrate the EPLSP approach with an SME case study on a Retail Management System (RMS) named FOCUS. Further, we leverage an evaluation framework to assess the effectiveness of EPLSP when applied to FOCUS. This case should define clearly the preferred areas of agility interference in the SPL, and where we need architecture to provide a sustainable production.**

Keywords- Enterprise Unified Process; Agile Unified Process; Software Product line.

I. INTRODUCTION

Modules, objects, components and services are all different patterns of the reusability practice. Software Product Line (SPL) is recognized as an approach for systematic reuse [1]. SPL matches software with different industries representing it as a manufactured tangible product. Further, it is one of the most important practices in sustainable organizations for the ultimate cost and time reduction [1].

SPL as an effective reuse approach is highly recognized in software enterprises. Small and Medium Enterprises (SMEs) do not firmly apply principles, but one can still recognize a chaotic version of such principles over their determined or formal processes.

SPL consists of three main activities namely Core Asset (CA) Development, Product Development (PD) and Management. CAs represents the basic reusable components in the SPL. CAs could be a class, a blueprint, a series of programming code or even a document, while the PD provides the means of final customer usable product. SPL management activity plays critical role in coordinating, supervising, planning and other administration practices needed across the production activities.

Agile methods promote productivity and values of iterative development over heavy-weight methodologies through number of practices that enable cost effective change [2]. Agile and SPL merge of practices covers the increasing need for shorter time to market and higher product quality [7]. On the other hand, the more the SPL becomes agile, it loses some of its essential properties, as strategic, planned reuse which yields to predictable results. The SPL reuse practice requires precise support in different areas like organizational capabilities, management and technical roles, architecture optimization…etc seeking a systematic approach for reusability. Incorporating agile practices in developing SPL raises some questions like what is the extent of interfering between the agile and SPL? And could agile fit in both CAs and PD?

SPL complexity promotes the need for an up-front design and heavy architecture [8]. CA development should conform to some standards and include detailed description and using instructions even if this CA is a Commercial Off-The-Shelf (COTS) component.

In this paper, we propose the Enterprise Product Line Software Process EPLSP as a roadmap for the implementation of the SPL with integration of agile practices. EPLSP covers the essential architectural practices in CA building, to solve the asset management pitfalls, and the use of agile practices in the PD to enhance the time to market variables.

EPLSP integrates the Enterprise Unified Process (EUP) [9] with the Agile Unified Process (AUP) [10]. EUP is an extension of the IBM Rational Unified Process (RUP) [11]. AUP is a simplified version of the IBM RUP that applies agile techniques in modeling, development and management [10]. Using the EUP overcomes the problems of managing such a family of products; like change management, strategic reuse…etc. EUP enables the enterprise to apply the

 ISBN: 978-1-61208-165-6

governance practices and disciplines (project management, retirement management…etc.) within the process. AUP allows for exploiting the agile essence to lighten the response to market requirements needed to enhance productivity. Further, AUP enables the customization of the development process to multiple agile processes or some of their combinations like SCRUM and XP. EPLSP focuses on the extent of agility needed in the SPL practice and where agility best fits in the SPL development life cycle. Further, EPLSP depicts where SPL could most benefit from its goals in the production level.

The rest of this paper is structured as follows: Section 2 surveys the state of the art in integrating agile practices into SPL. Section 3 depicts the EPLSP process and the artifacts produced in each step. Section 4 demonstrates EPLSP on the Retail Management System (RMS) FOCUS. Finally, Section 5 concludes the paper with remarks for future work.

II. RELATED WORK

Investigating whether Agile and SPL could integrate to complement each other; there stills a debate among the research community about its extent and feasibility.

Tian and Cooper [2] argue that the combination of Agile and SPL forming the Agile Software Product Line Methodology (ASPLM) could shorten time to market maintaining the quality, in which the ASPLM leaves room for futher development work to meet customer's changing requirements, rather than pure customization of CA. They showed that CA, PD and SPL Management activities need to be investigated for possible agility.

Carbon et al. [3] had conducted a class-room experiment following the motivation to present preliminary results showing the successful merge between Agile and SPL. They concluded to a result that agile in SPL reduces time spent on design (Increases the speed), while SPL keeps changes to minimum (Increases quality).

On his research, Geir K. Hanssen [4] stated an answer for how to combine Agile and SPL. In a successful marriage, he stated that this combination leads to; risk reduction, organizational development, reduced maintainability, community building, openness and visibility and company culture improvement, contributing to the emergence of a software ecosystem, which refers to how organizations should exist together as an ecosystem.

One of the popular case studies conducted by the Software Engineering Institute in Carnegie Mellon University is Salion [5]. Salion is an SME with no experience in its application area. It pursued a reactive approach to its Agile SPL achieving a phenomenal reuse level of 97% with its 21 employees counting seven developers only.

Despite the success of the previous cases, they did not take in consideration the difference in nature between the CA and the PD. As any other production the sustainability of the production depends on the systematic the whole process, which should be only achieved by architecture

III. ENTERPRISE PRODUCT LINE SOFTWARE PROCESS (EPLSP)

We propose EPLSP as a software process with the goal of effective production of SPL that better meets its market requirements. EPSLP integrates agile and SPL practices from the two extensions of IBM RUP namely EUP and AUP. EPLSP covers the Enterprise disciplines needed in the SPL to improve the change management and architectural variability in the CA phase. These parameters are improved while taking into account the increasing demand on lower time to market and quality software production through employing agile practices.

A. EUP and AUP

EUP is an information technology lifecycle that encompasses the activities of an IT department. Further, EUP adds the enterprise disciplines required to effectively manage organizations' portfolio of systems as described in Figure 1.

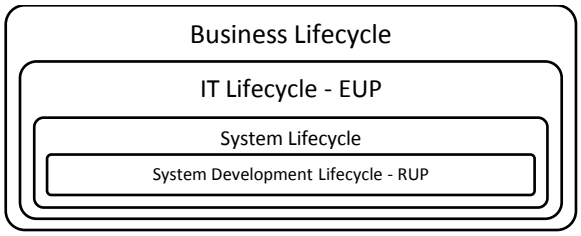

Figure 1. The Scope of different process lifecycles.

EUP extends RUP to include the operation and support of a system after being in production along with its eventual retirement, where the two new phases benefits the concept of strategic reuse promoted by the SPL. Further, EUP enhances the overall process with the separation of the disciplines into; development, support and enterprise as illustrated in Figure 2.

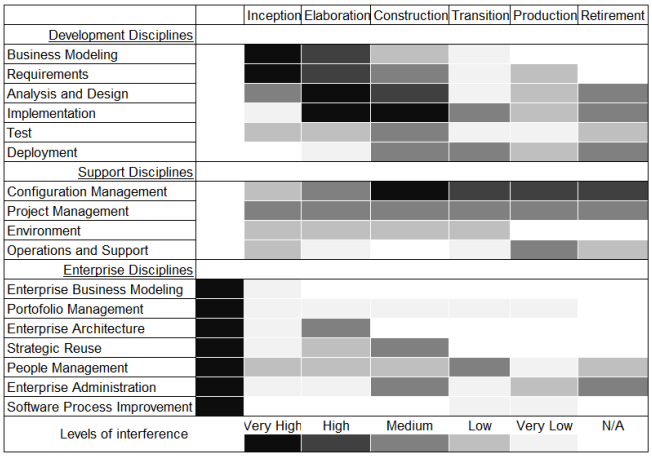

Figure 2: Enterprise Unified Process [9]

AUP is an Ultra-lightweight variant of RUP, with the work disciplines and products simplified and reduced as shown in Figure 3.

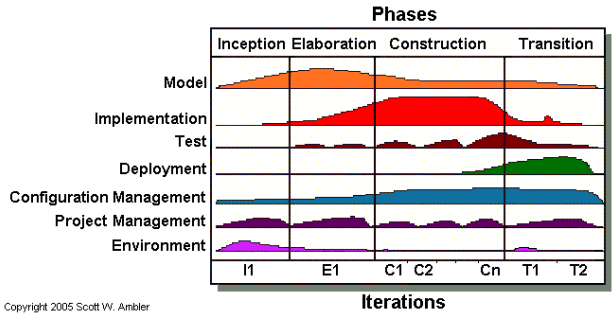

Figure 3: The Agile Unified Process.

We employ the practices of EUP and AUP that facilitate different management levels and all involved parties in the production activities to highly control tasks associated to their roles. Those practices complement the EPLSP and close the IT department circle within a tightly managed manner with the following recommendations;

- Documenting architecture using Unified Modeling Language (UML)
- Applying SCRUM as an Agile project management practice
- COTS could be used across the product line
- Configuration Management software is essential to manage releases
- Specific software to manage commonality and variability to enhance the strategic reuse option.

The different nature of the CA and the products is one of the major challenges facing the application of EUP and AUP to SPL. This marriage between EUP and AUP is intended to facilitate the application of both processes to SPL. CA needs the architecture provided by the EUP and the extension of the production and retirement phases. The need for fast response to market for the products could be achieved with agility. AUP has the same phases as EUP but simplified, so there is no need to rework the architecture of the artifacts to fit in the other SPL production activities.

B. EPLSP Process

EPLSP provides means to integrate agile practices into the SPL development life cycle. Figure 4 depicts the overall process structure in EPLSP. The initial phase on the bottom of the process consists of the domain engineering, in which it represents the knowledge needed to build the reusable artifacts like; scoping, requirement engineering, design, testing, and the realizing of the commonality and variability of the product line practice with the CA development activities. In the middle there exists the CA base which contains the reusable artifacts. The right downward arrow represents the reactive approach in which the start point is the PD.

The PD activity is split into two tasks, development task and release task for two reasons, the separation between the deployment and the production which differs in the application of disciplines, and to maintain a direct agile incremental iterative practice.

The management tent could be seen as the containing rounded box, providing SPL process with the needed management disciplines solely.

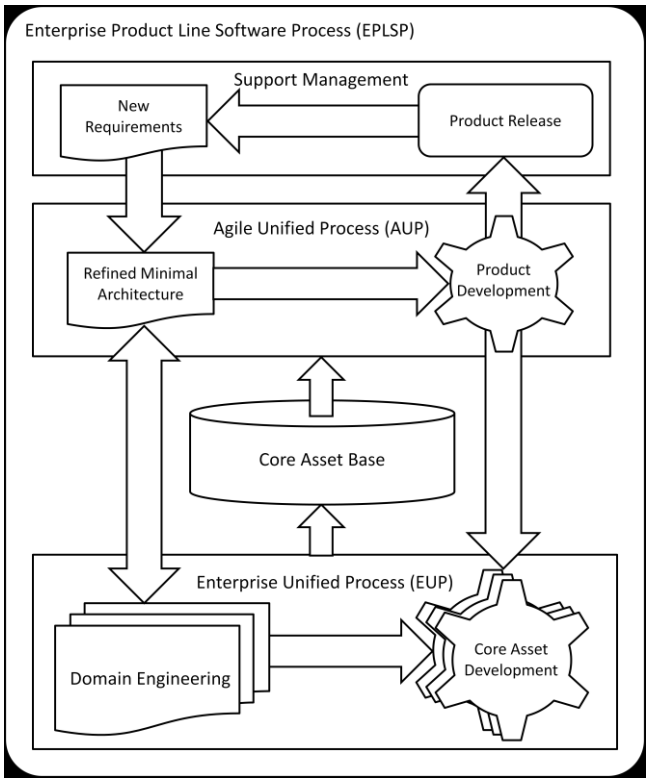

Figure 4. EPLSP Conceptual Model

CA development is the activity intended to build the reusable components of the SPL. CA development requires prior domain expertise, heavy architecture and management capabilities. This could be achieved only by a well defined engineering architectural centric process to ease the reusability of this asset. EPLSP proposes the application of the EUP as a basic process for the domain engineering and CA instantiation as shown in Figure 5.

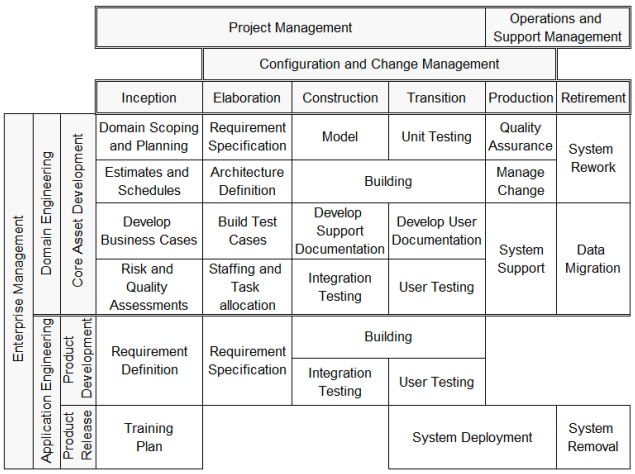

Figure 5. EPLSP Milestones

PD activity is usually in need of the fast response to customer requirements, and early delivery of quality products. These goals could be achieved by the agile methodologies, for this reason EPLSP preferably uses AUP as a simplified version from the unified process to eliminate unneeded heavy architecture. Figure 5 determines milestones in every phase of the EPLSP.

IV. FOCUS® RMS

This section describes an RMS named FOCUS to demonstrate the feasibility of the EPLSP process. Further, we discuss FOCUS commonalities and the challenges we faced during and after the development process.

A. *FOCUS® subsystems:*

FOCUS® is a mini ERP specially developed for small and medium retail outlets. This system could work as one unit, integrated and linked over one database or every subsystem separated as a single unit as depicted in Figure 6.

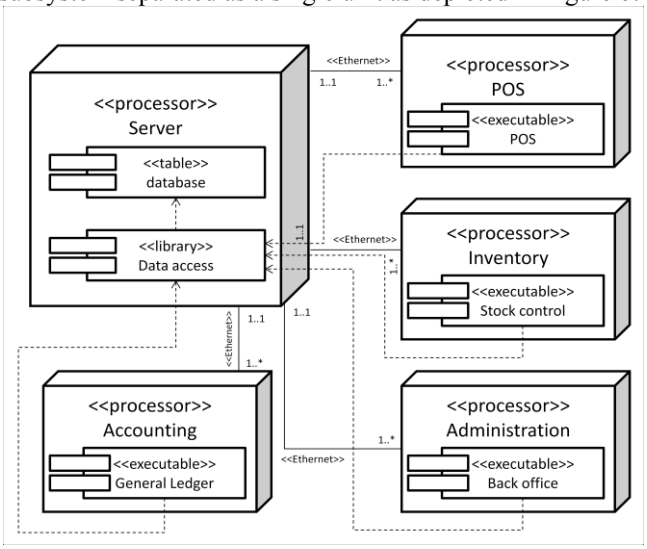

Figure 6. FOCUS® RMS Deployment Diagram.

FOCUS is composed of the following subsystems:

- FOCUS® stock control, which holds the essential stock transactions; basic entries, receiving, item cards…etc.
- FOCUS® Point of Sale (POS): is where daily sales transactions managed by salesperson in the checkout area of an outlet or a shop.
- FOCUS® General Ledger (GL): reflects automatically the daily selling, receiving and monetary transactions to journal entries and accounts, and reports financial statements.
- FOCUS® back office is the administrative tool, which facilitates higher management to monitor transactions, authorize permissions, link subsystems and modify system settings.

The system was primarily developed to target large sector of retail outlets with the following features; installed, not customizable, self setup with a simple instructions guide and easy to understand and apply. Since these requirements could rarely be found in SME's business software, it was planned to produce enhanced version yearly with new features; based on wide survey for user requirements.

Figure 7 depicts the system requirements and demonstrate the similarities as classes, layers and complete sub modules; like the security module, transaction file and product catalogue.

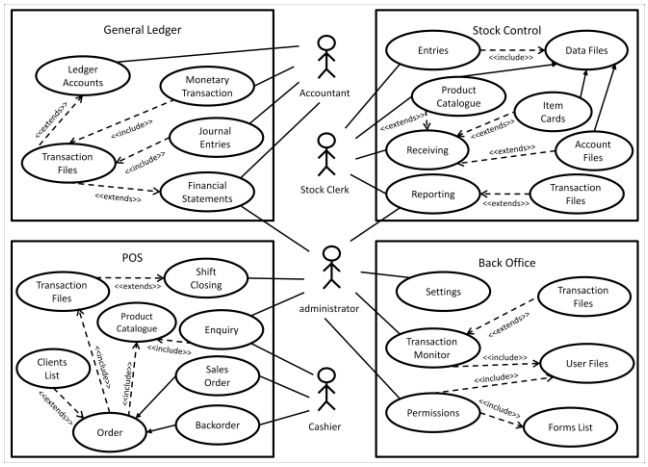

Figure 7. FOCUS® RMS System Requirements.

1) *Company*

The software was built in a small enterprise named SCOPE Communications, in which it employs 13 people; 6 only is counted as developers, and it took 18 months to release the basic version of the full system.

This basic version of the system contains 135 KLOC in total, with 160 database tables, 1100 stored procedures, 450 forms and 320 reports covering the four modules.

The core process was a simple version of the incremental, iterative process; it was described and documented using the UML. The system was built using a similar proactive approach to the SPL's, with no use of any Configuration

Management software. Test cases are prepared with two concerns; business cases depend on customer stories and technical cases over the functions, for data integrity.

2) FOCUS Production Challenges

From our study of the former system we have observed some challenges resulting from the application of the previous process, like;

- Recurrent costs associated with the reuse of non-architectural artifacts
- Higher risk resulted from unplanned resource allocation and estimation
- Complexity of managing the commonality and variability of artifacts
- Wasted time resulted from the duplication of code and documentation
- Corrective bug fixing rather than preventive associated with the unplanned test cases
- Customers frequent complaint from support

3) EPLSP and FOCUS

Appling EPLSP to FOCUS RMS will help the company well manage the SPL process, with the allocation of the architectural centric activities in the needed areas only; which is intended to well manage changes across the process, and the use of agile practices in the PD activity to improve the market response.

4) Refactoring FOCUS®

As a retail management system the product catalogue regarded as the main component in the solution, therefore; the selected artifact to be redesigned using the EPLSP is the product catalogue, which contains the building features of any product like name, description, type, category, price, etc.

The product catalogue is considered a sub module, and is completely used in one of the main modules, and partially used in the three other modules.

5) Applying EPLSP to FOCUS

The product catalogue features totally differs as the type of products or services provided by the outlet itself, however there are some common requirements in this sub module.

The architecture definition in the EPLSP elaboration phase defines a practice to manage the commonalities and variability of the product catalogue. This covers the change management problem and reduces the recurrent costs resulting from unplanned reusability.

The main goal of the EUP unique production phase is to keep systems useful and productive after deployment, in which it encompasses the operation and support of the system. Also, this phase provide some means of quality assurance by monitoring the operation of the system when working and recovering any problem. These practices help the company manage the post deployment stage professionally, which develops customer loyalty.

We are redeveloping the product catalogue as a sub module with EPLSP maintaining the same functionality of the catalogue. We compare the development experience using EPLSP with its counterpart using the older version of the system developed with an iterative simple RUP. The metrics used for our comparison are depicted below in subsection 6.

The product catalogue itself consists of two parts. One part is recognized as a core asset, which includes the search base and the basic entry forms like category, product, limits…etc. The second part is realized as a product which includes product labeling, reports…etc.

We develop the product catalogue core asset using EUP as the part of EPLSP that incorporates a complete architecture, while developing the product part using SCRUM. In both parts we use an incremental iterative process.

In the older version of the FOCUS system, we employed a simple iterative and incremental undefined process to develop the whole SPL. The sequence of the process steps mostly relied on the task, the feature or even on the developer. The older process employed code comments and traditional UML diagrams for documentation.

Using EPLSP, we define 5 essential practices. We use a tailored version of SCRUM at the product part of the catalogue and a set of architectural templates and plans in the CA part. Further, we utilize configuration management software and a set of chosen UML diagrams for core assets and the products. We define the development incremental steps as shown in Figure 8.

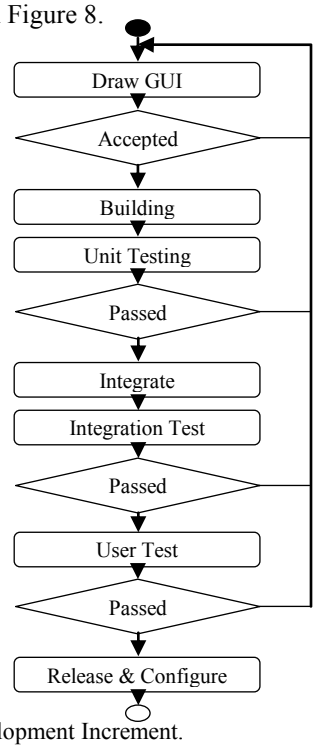

Figure 8. Development Increment.

For the product part of the catalogue, we define a set of SCRUM roles namely the project manager, the product owner and the developers. The project manager acts as the SCRUM master, while the marketing team acts as product owners. Further, we have a team of a senior developer and two junior developers.

A daily meeting is held with the team to discuss the progress and the problems. Further, a weekly meeting is held with the presence of the product owner to present the features achieved thus far. The weekly meeting aims also at collecting feedback from the product owner while developing new ideas and requirements. Finally, a monthly meeting is held to test and show the released version, which could be installed at the customer site for free. Such installation allows the support team to record comments within two or three days.

In the production phase, the product backlog is developed in cooperation between the SCRUM master and the product owners. The product backlog scenarios are prioritized while dependencies are identified. Further, the product backlog is revised and updated in every monthly meeting. The sprint backlog defines the current set of features in the construction phase, its tasks and associations to team members. These backlogs contain:

- Use cases, Class and Activity diagram.
- Test cases.
- Schedules and job orders.

Finally, the released version of the product is configured and generated with a set of user instructions.

We produce the following set of architectural documents during the development of catalogue CA part. Such documents contain the complete domain architecture that depicts the infrastructure CAs. Infrastructure CAs include the CA part of the product catalogue along with other CAs :

- Detailed business case.
- Requirements and specifications plan.
- Test plan for the 3 testing levels, unit test, integration test and user test.
- Software development plan
- Iteration plan.
- Change and configuration plan.
- Deployment and support plan.

Unlike the product development, the configured version of the core asset is augmented with the developer's manual and deployment instructions.

6) Process Validation

We utilize a number of metrics to assess the effectiveness of EPLSP and compare it to the classical iterative or incremental development process. These metrics are defined to assess the effectiveness of the merge between SPL development and agile process and it was stated and used in Salion's Agile SPL [6] as follows:

- Reusability: Salion [6] defines the reusability of its system with a percentage level that is equal to common files used in all members of the product family divided by the total number of files generated across the product line (Reusability level% = common files/total SPL files).
- Time to market: It was proposed in the same case [6] as the manpower used per month to produce the first customer's product (# of persons-month).
- Eliminating duplicates: We measure it by the percentage of eliminated duplicates using the classic Line of Code (LOC) metrics (Eliminated Duplicates% = # of duplicated LOC/total LOC).
- Productivity: This metric is measured using popular LOC and Use Case metrics as an extension of the Function Point metrics as a complex subject concerning a relation between different resources or artifacts, the use case metrics defines an early – prior development measure of software functionality rather than the function point, which could only be used after development.(Usecase/hour, LOC-person/month…etc)
- Cost reduction: Similar to of the productivity metrics, but it is preferred to be measured by the Use Case metrics. Also either LOC or Function Point could be used, but regarding the LOC it will be subjective due to the difference in number of produced lines from one person to another within the same class. And for the function point analysis it could be determined only after the development completion; instead of early determination of cost in the case of Use Case metric.(UseCase-person/day)
- Defect Removal Efficiency (DRE): Is one of the popular quality metrics which is intended to measure the discovered errors during development in relation to the total errors and defects found.

  (DRE=E/(E+D) in which E is the number of errors and D is the number of defects).

V. CONCLUSION

This paper proposed EPLSP to address the possible integration between SPL and agile. Applying this process to FOCUS RMS addresses most of the challenges the company faced during the production of the software using the classical process. Further, the proposed EPLSP addresses the time to market challenge, which is one of the major SPL challenges. EPLSP addresses the challenges through leveraging agility in the suitable areas of integration of the EPLSP which helps the production quality software products.

Applying EPLSP to FOCUS RMS, our potential challenges include technical and social challenges. Technical challenges include training the development staff in the EPLSP development process and reworking the design. Our social challenges confine the commitment of the upper management to change and restructuring the organization so that the new process is accommodated.

 ISBN: 978-1-61208-165-6

REFERENCES

[1] Linda Northrop. 2008. Software Product Lines Essentials. *Software Engineering Institute, Carnegie Mellon University*. http://www.sei.cmu.edu/productlines/frame_report. [accessed, April 2011]

[2] Cunningham W, Manifesto for Agile Software Development. *2001. [cited 2008-09-30];* Available from: http://www.agilemanifesto.org. [accessed, March 2011]

[3] Tian, K. and K. Cooper, Agile and Software Product Line Methods: Are They So Different?, in *1st International Workshop on Agile Product Line Engineering.*2006.

[4] Carbon, R., et al. Integrating Product Line Engineering and Agile Methods*: Flexible Design Up-front vs. Incremental Design*. in *Workshop on Agile Product Line Engineering*. 2006.

[5] Hanssen, G.K. and T.E. Fægri, *Process Fusion - Agile Product Line Engineering: an Industrial Case Study*. Journal of Systems and Software, 2007, pp. 836-849.

[6] Clements, P. and Northrop, L., *Salion, Inc.: A Software Product Line Case Study* , Software Engineering Institute (SEI) Technical Report CMU/SEI-2002-TR-038, Carnegie Mellon University, Pittsburgh, PA, November 2002.

[7] Snorre Gylterud, Constructing a Silver Bullet? *Combining Software Product Line Engineering and Agile Software Development, A thematic literature review*, Norwegian University of science and technology, *2008*.

[8] J. Bosch, *Design and use of software architectures: adopting and evolving a product-line approach*. Addison-Wesley, Harlow, 2000.

[9] S. W. Ambler, J. Nalbone, M. J. Vizdos, The Enterprise Unified Process, *Extending the Rational Unified Process,* Prentice Hall*, 2005.*

[10] S. W. Ambler, The Agile Unified Process (AUP), *Ambysoft, 2005*; www.ambysoft.com/unifiedprocess/agileUP.html. [accessed, March 2011]

[11] Philippe Kruchten, *The Rational Unified Process: An Introduction*, 2nd ed. Addison-Wesley, 2000.

[12] Rubin, H. A. "Macro-Estimation of Software Development Parameters: The ESTIMACS System." *Proc. SOFTFAIR: A Conference on Software Development Tools, Techniques, and Alternatives.* New York: IEEE, July 1983, pp. 109-118.

 ISBN: 978-1-61208-165-6