A Secure Framework for Software Product Line Development

Article *in* International Journal of Computer Applications · February 2017

DOI: 10.5120/ijca2017912919

3 authors:

Md Mottahir Alam
Singhania University
15 PUBLICATIONS **28** CITATIONS
SEE PROFILE

Asif Irshad Khan
King Abdulaziz University
31 PUBLICATIONS **87** CITATIONS
SEE PROFILE

Aasim Zafar
Aligarh Muslim University
38 PUBLICATIONS **39** CITATIONS
SEE PROFILE

A Secure Framework for Software Product Line Development

Md. Mottahir Alam
(Ph.D. Scholar)
Dept. of Computer Science,
Singhania University,
Jhunjhunu,
Rajasthan, India

Asif Irshad Khan
Dept. of Computer Science,
FCIT,
King AbdulAziz University,
Jeddah, KSA

Aasim Zafar
Dept. of Computer Science,
Aligarh Muslim University,
Aligarh, UP,
India

ABSTRACT
In today's marketing scenario, the software companies have the challenge to provide a vast variety of customized software products option to satisfy diversified customers' requirements. Although increasing product varieties increase sales volume and profits, but it also raises development complexity, time and cost. In order to address the issues, companies are moving towards Software Product Line Engineering (SPLE) which helps in providing large varieties of products with minimum development effort and cost. This approach amalgamate component based development and feature based development, both of which are based on the concept of reusability and facilitate the development of a family of products. This paper tries to propose an improved framework for software product line. Cross-cutting concerns such as security and configurability are addressed in this framework. Further, the proposed framework is compared with selected state of art frameworks.

General Terms
Software Product Line Engineering, Variability Management.

Keywords
Software Product Line, feature coverage, variability, comparison framework, product line methods, feature modeling.

1. INTRODUCTION
Although McIlroy [1] in 1969 and Parnas[2] in the 1970s were the first to describe the software product lines and its benefits, it started receiving special attentions in software engineering community since the 1990s [3,4,8]. Recently, Software product lines have significant attention in both research and industry because software product line engineering(SPLE) provides a promising method to deliver a vast range of improved quality, cheaper and faster[3] software systems.

SPLE technique is based on mass customization[12] to build a set of products or systems that have a commonly managed set of features and are developed from a common set of reusable core assets in a way to fulfill needs of a specific customers or market segment[3]. Therefore, software product line(SPL) is accepted as an effective approach for reuse in software engineering[9].

The main advantages of the product line are reduced time-to-market[10,11], reduced cost[12] and improved quality [11,12]. When SPLE is combined with component based development approach, it further increases the scope of reusability, resulting in further reduction in release time and cost without compromising on the quality of the product[5,6,7]

As a result of this, several software companies have already switched or considering switching to the software product line approach [13,20] with an incorporation of components into it.

The products in SPL are developed on a common platform by binding variability on top of that platform. The use of platforms in application development is possible through planning reuse, building reusable assets, and proactively reusing these assets. The reuse repository of a software product line is known as core assets of the product line. It includes domain models, requirements, architecture, components, test cases, support tools etc.

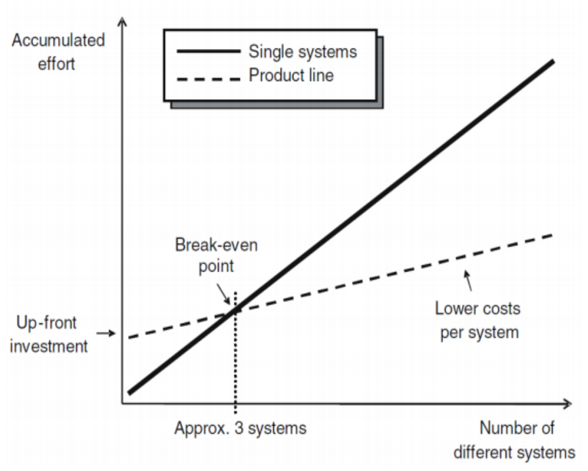

Fig 1: Economics of software product line engineering [17]

Fig 1 compares the economics of SPL with that of traditional software development. As we can observe from the graph, although SPL involves a large initial investment in time and cost[17] compared to traditional software development, but it maximizes return on investment(ROI) by reusing architecture and other core assets across the product line family. Furthermore, it also helps in achieving maturity of architecture and software development processes.

The three important conceptions in SPLE are commonality, variability, and configuration. Commonality refers to common aspects in all applications whereas variability indicates the application-specific features in SPL. Configuration comprises of the selection of possible variants for a specific application.

In today's competitive market, although there is steep competition to release products faster, there should be no compromise on quality and stability of the products[19].

SPLE ensures improved quality products along with reduction of time to market. Apart from this, it also helps in lowering of maintenance cost, improving cost-estimation, and offering customized products to customers. Furthermore, it enables evolution of wide range of newer improved products.

This paper is organized as follows: Section 2 discusses related work in software product line framework. Section 3 discusses the proposed framework. The comparative analysis of frameworks is presented in Section 4. Section 5 presents discussion and section 6 provides concluding remarks and future work.

2. RELATED WORK

Kim, J. et.al[14] in 2008 proposed a frame implementing domain requirements as well as modeling core architecture in SPL. It is presented in Fig 2. The framework provides a mapping between product line requirements and reference architecture through the use of processes, methods and support tools. It involves concepts such as goal oriented domain requirement analysis, analytical hierarchy process(AHP), matrix technique and architecture styles. It performs domain requirement analysis by classifying requirements into four abstract levels:

Business level, service level, interaction level, and internal level. This helps in identifying and building components. The next step is to prioritize the components using matrix techniques and analytical hierarchy process(AHP). Finally, a reference architecture is created based on the components and their quality attributes.

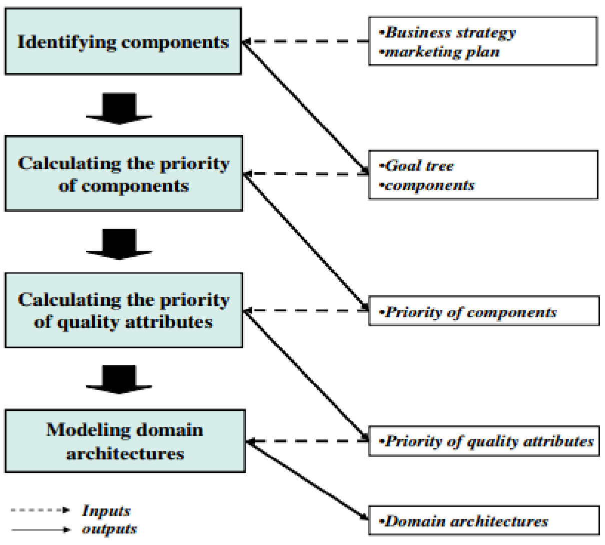
Identifying components
Calculating the priority of components
Calculating the priority of quality attributes
Modeling domain architectures
•Business strategy
•marketing plan
•Goal tree
•components
•Priority of components
•Priority of quality attributes
•Domain architectures
Inputs
outputs

Fig 2: DRAMA process[14].

Tanhaei, M. et.al[15] in 2010 proposed an architecture-based technique to select constituent components in an SPL. It is a component-oriented technique to manage and control the selection of components in an SPL, thereby reducing risks and cost of software development. The components are carefully selected on the basis of reference architecture, product family requirements, domain requirements, and concerns of stakeholders. The architecture of this method is shown in the Fig 3. It starts with the selection of a component list from the component lists on the basis of architecture variant point.

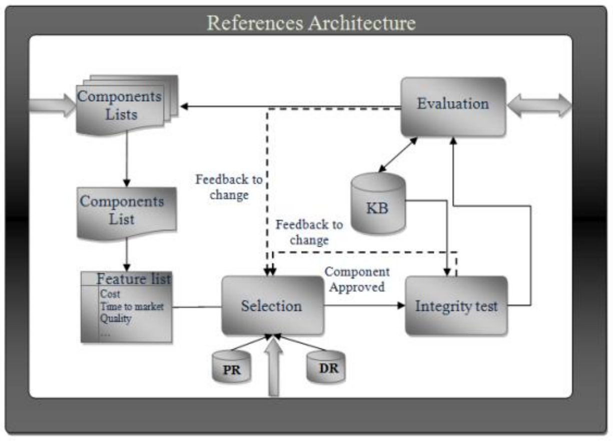

Fig 3: Architecture of method[15].

The components in the component list can be selected either from COTS components or component repository. If the component is not available then it is developed. The selected components are evaluated for approval. Once approved, these components are passed through integrity test. Lastly, a reference architecture is obtained these selected and successfully tested components.

Mellado, D et.al[16] proposed a framework incorporating security mechanisms for SPL as shown in Fig 4. This framework divides activities into two main types: application engineering and domain engineering. It implements security by integrating domain security mechanism PLSecDomReq and application security mechanism PLSecAppReq.

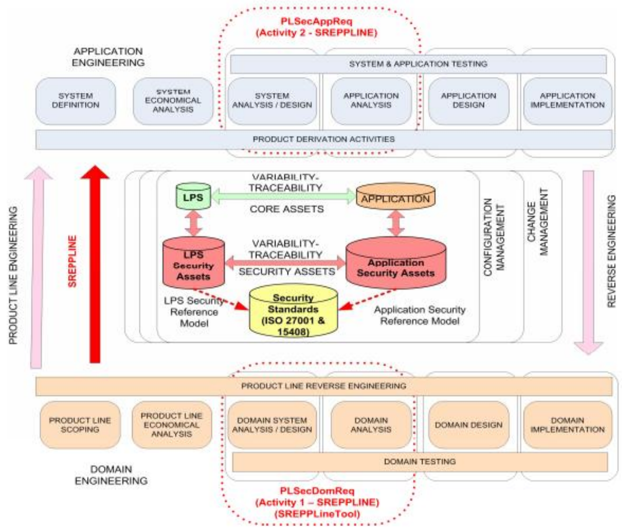

Fig 4: Security requirements engineering framework for Software Product Lines [16].

The reference architecture involves repositories, traceability, and security mechanism. The various repositories implemented in the framework are:

i) Software product line repository

ii) Application repository

iii) Software product line Security Assets repository

iv) Application Security Asset repository

v) The security standards repository

Guendouz, A et al.[18] in 2014 proposed a component based approach for SPLE and validated it through a case study of e-Meeting. The main benefit of this step is the automation of application derivation step. The framework is presented in Fig 5.

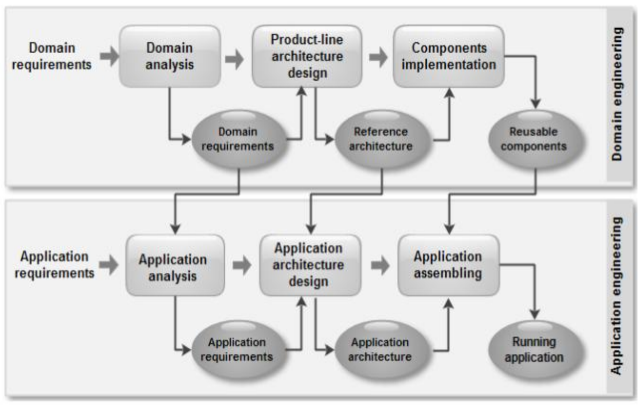

Fig 5: Component-Based Product Line engineering[18]

3. PROPOSED MODEL

The proposed model as shown in Fig 7 represents a high abstract level of software product line (SPL) architecture. The model is a mix of aspect-oriented and the feature-oriented approach. The aspect-oriented approach addresses crosscutting concerns and functional behaviors of SPL while feature-oriented approach is used to capture variability and commonality of product lines. The detailed explanation of the proposed model is as follows:

The model has 2 high-level processes: domain engineering and application engineering. The main aim of domain engineering is to identify and develop reusable artifacts for reuse later in the application engineering phase. Application engineering targets building of software products using the identified reusable artifacts.

Domain Engineering requires common and variable requirements of the product line family as inputs and generates reusable core assets such as components, framework, a library, tools or a platform etc.

Application engineering deals with requirements specifications of individual products of the software product line family are considered and a customer-specific product is developed by using the generic architecture and reusing the core assets from domain engineering as much as possible.

3.1 Domain Engineering

The core activities of the domain engineering phase are described as follows:

Business Feasibility Study:

This activity is concerned with research and analysis of all the factors concerning with the success of a new set of products in the marketplace. It requires the active participation of all possible teams such as software development team, marketing teams, business investors, buyers and prospective customers.

The development team will give the technical and resource perspective while the marketing team, buyers, and end-customers will help to bring a business perspective to the analysis. It involves the gathering of business intelligence, competitive studies, and assessments, etc. and then combining of all of these data into a solid business strategy and plan.

A feasibility study is conducted in terms of benefits and risks of the target product line family. The benefits of a product line family are the returns that are expected from it while the risks are the threats that are expected due to its introduction. The comparative study of benefits and risks gives a broad picture to the business stakeholders which enables them to take an investment decision.

Product Line Scoping

It is based on the previous step in which the features of the potential product line and its products are identified. The scope should be practically attainable .i.e. it should neither be too large nor too small. The output of this activity is a product portfolio comprising of all potential products of the product line family and also a product roadmap as shown in fig 6.

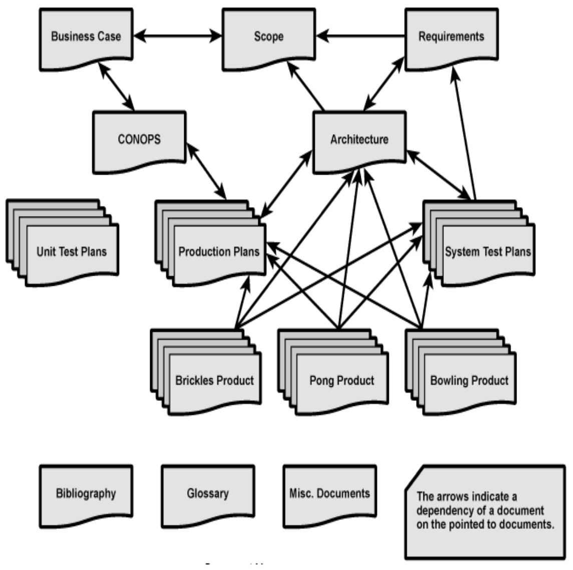

Fig 6: Product Line Scoping [21]

Product Line Requirement Analysis:

In this activity product line requirements are analyzed and documented. Domain experts and different stakeholders of the product line are involved to analyzed the variable and common feature requirements targeting the family product. Core reusable artifacts are identified and suggested for later reuse. The requirements are categorized into a set of reusable common artifacts and product-specific variable features for the product line family. It is essential that product line requirements should be appropriately structured and should be free from design and implementation assumptions.

Security Policy and Security Modeling:

Security is across-cutting concern and is affected by a wide range of architectural decisions. Since a product line also contains third party components, therefore product line architecture should be able to contain the potential security concerns posed by these components.

Based on the thorough understanding of product line's security requirement, a security policy for the product family is defined. The security policy generally covers about authentication, authorization, integrity, confidentiality, and the handling of data. Additionally, it acts as a foundation for making access control decisions by identifying different privileges a user should have while accessing the secured assets.

A security architecture language such as XACML(eXtensible Access Control Markup Language) or Secure xADL (secure eXtensible Architecture description language) is used to describe the security requirements. These policy rules are formally described into a security model so as to validate the product line against it in the later stages. The security model is built considering all the scenarios representing the security requirements for the proposed product line. It includes vocabulary for security requirements.

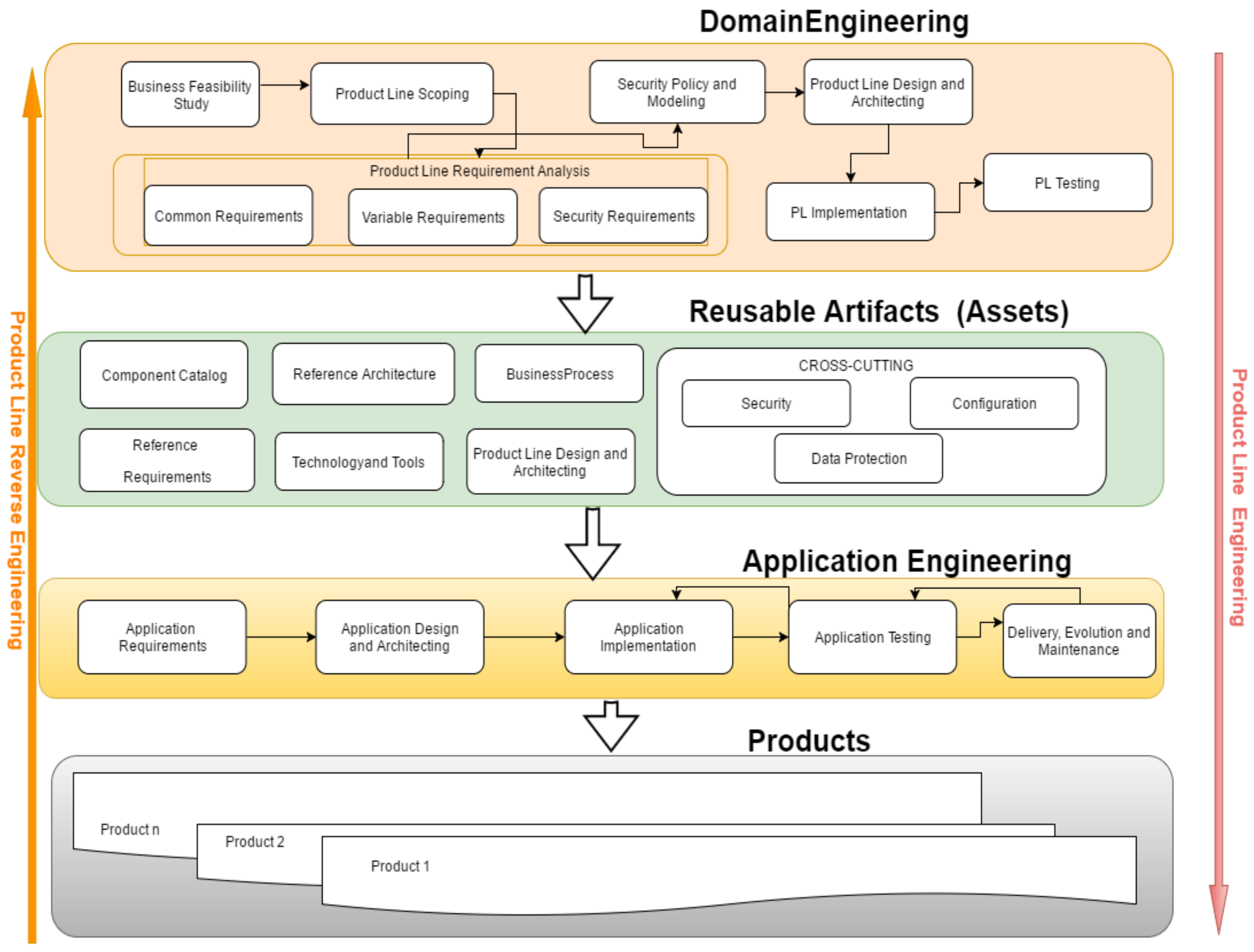

Fig 7: Shows a high abstract level of Software Product Line (SPL) Architecture

The security policy rules should be in adhering to international security standards such as ISO/IEC 27001 standard, ISO/IEC 27002 standard, and ISO/IEC 15408-1:2009 standards.

Product line design and architecting:

Product line design refers to designing the basic reference architecture as well as deciding the architectural style for a software product line. It describes all the mandatory and varying features of the SPL domain and provides a design which is fundamental to all products in the product line family.

The reference architecture of the product line is defined on the basis of these features. It also provides room for the expression of variability and commonalities of the product instances and helps in creating an abstract structure for the product set. The products are abstracted by configuring the architecture and tailoring components available in the component catalog.

Many related product architectures are encapsulated into product line architecture. It is like a common architecture for a set of related products in the product line. Product line architecture(PLA) defines the diversity of the product as each product architecture varies from product to product but adapted same reference architecture of the product line. The variation points in a domain artifact identify the places at which the products differ or variation occurs.

Architecture description languages (ADLs) provide support for capturing variation points. ADL is recognized as an important element in the description and analysis of

software properties. It allows describing of variable and dynamic features in the SPL. Koala and xADL 2.0 are some of the ADLs which provide exclusive support for capturing variation points. However, not all ADLs provide such type of support.

Product line Implementation:

It involves finding of available services such as components, and tools existing in the local or remote repository on the basis of the specifications of the design phase. It also includes detailed designing and realizing of the reusable software components for the entire product line family.

It describes a set of activities and supporting tools essential for building the products. It also elaborates about implementation plan which consists of policies, procedures, automation support for reuse during application engineering phase.

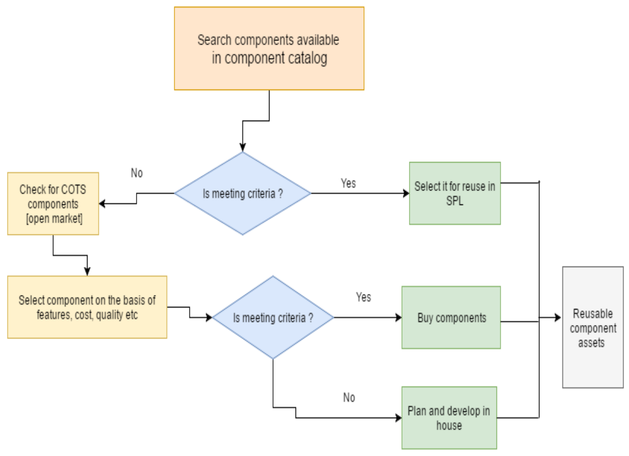
Search components available in component catalog
Is meeting criteria ?
No
Yes
Check for COTS components [open market]
Select it for reuse in SPL
Select component on the basis of features, cost, quality etc
Is meeting criteria ?
Yes
Buy components
No
Plan and develop in house
Reusable component assets

Fig 8: An approach for component selection

The framework provides an approach to select components as shown in Fig 8. The approach consists of systematic set of activities. The component selection starts with searching of suitable components in the locally available component catalog. If suitable component meeting all the specified requirements is found, then it is selected for reuse. In case, the required component is not locally available, then one should search in market for component with required features, preferably in the same domain.

These externally available components should be selected on the basis of features, cost, quality etc. If any of externally available components meet all the criteria than it selected. In case, no component meets the criteria then one should plan and develop own components with desired functional and non-functional features.

Product line Testing

Software product line brings enormous testing efforts. It is quite challenging to possible test all family products. Validation is the process of carefully checking the requirements for completeness, exactness, clarity and uniformity. In software product line development, requirements' validation occurs in stages and it has a large number of reviewers.
Test artifacts like tools, test case, test data, metrics are generated in this activity to be reused during application testing phase in order to test a chosen system configuration.

3.2 Application Engineering Phase

In application engineering phase, new products are created by reusing the core assets developed during the domain engineering phase. The common requirements are taken from the domain engineering and product-specific variable requirements are added to bring out new products. Thereafter, the products are rigorously tested and if they meet the acceptance criteria, can be delivered.

Following are the detail explanations of the major activities carried out to create customize products:

Application Requirement

The main focus in this phase is to derive customized product's variants as per stakeholders' functional & non-functional requirements. Feature Profile (FP) of a specific product is generated starting from its specification documents. Features are prioritized based on the preferences of stakeholders and business objective concerning the optimal features and quality needs.

Specific extraction rules are defined for analyzing the specification documents in order to identify the common and variable features for each product.

Application Design and Architecting:

Product design and architecting refers to designing the individual product architectures for a set of products in a product line family. It avails the reference architecture from the domain engineering phase to bring out the product architecture. It picks and configures the compulsory parts of the reference architecture and adds in product specific variations.

The inputs to this step are reference architecture and the product requirements' specification while the output is product specific architecture for individual products. It is to be noted that the product(application) architecture is not developed from the scratch, but is extracted from the reference architecture by linking variability, i.e. making specific selections at points where the reference architecture suggests different variants.

Furthermore, product design and architecting must be done as per the rules defined in the domain design phase while building the reference architecture. These rules are defined for implementing variability as well as application-specific adaptations.

Application Implementation:

During the implementation sub-process, individual products are built, using selections and configurations of the reusable components as well as the implementation of product-specific features.

The inputs to the product implantation sub-phase are the application architecture and the reusable components from the component catalog while the output is a fresh product with the detailed design artifacts.

Application Testing

During the testing, the new product is verified and validated against its requirement specifications. Application testing includes all common parts, and variable parts in order to achieve complete test coverage.

The input to the product testing includes all kinds of products artifacts to be used as a test benchmark, the newly built product, and the reusable test artifacts from the domain testing.

The output of the product testing consists of a test report with the results of all tests that were carried out on the product as well as the detected errors. Additionally, the detected defects are documented in more detail in bug reports.

If the product acceptance criteria is not met, the process will loop back to the product implementation phase as shown in the proposed framework.

Delivery, Evolution and Maintenance:

The final phase of the application engineering phase is the successful acceptance and delivery of the final product. If the new product has been successfully tested, fulfilling the acceptance criteria, it can be delivered. If the product fails to meet the specifications, it has to be rebuilt and tested again.

Once delivered, the company works for product improvement and comes up with possible new product variants so as to remain competitive.

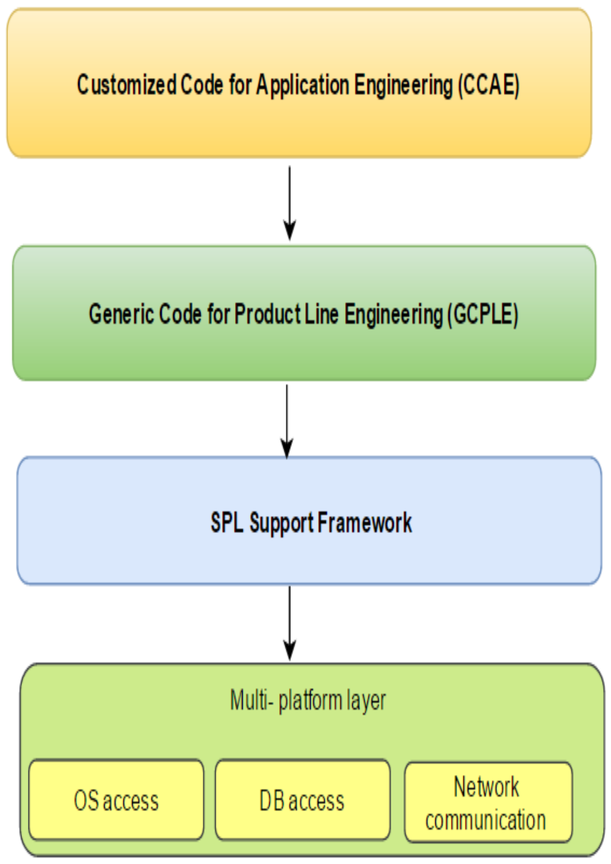
Customized Code for Application Engineering (CCAE)
Generic Code for Product Line Engineering (GCPLE)
SPL Support Framework
Multi- platform layer
OS access
DB access
Network communication

Fig 9: Layered architecture for SPLE development

Fig 9 shows layered architecture for SPLE development. GCPLE layer acts as a set of services for developing customized applications in CCAE layer. It uses the SPL support framework for generating code addressing the commonalities and variabilities of the product line. Multi-platform layer provides generic services such as operating system(OS) access, database(DB) access and network communication.

4. COMPARISON OF DIFFERENT APPROACHES

A comparative analysis of above approaches is presented in Table1.

Table1: A comparative analysis

Issues and Challenges	DRAMA Kim, J. et.al	Tanhaei, M. et.al	Mellado, D et.al	Guendouz, A et al	Alam, M et.al
Product line management					✓
Requirements	✓	✓	✓	✓	✓
Variability management	✓	✓	✓	✓	✓
Designing and Architecting	✓	✓	✓	✓	✓
Implementation	✓	✓	✓	✓	✓
Cross-cutting concerns			✓		✓
Testing & Validation		✓	✓		✓
Maintenance and Evolution			✓		✓

5. DISCUSSION

It should be noted that domain engineering is a continuous process. The knowledge and understandings concerning the domain should be maintained and continuously revised according to new experiences, scope broadening of the product line, and new trends. Furthermore, domain engineering should also be continuously adapted as per the feedback and experiences from application engineering. Thus, domain model practically can never be fully complete; it may perhaps always be refined to be more accurate.

In application engineering phase, reusable assets such as components, interfaces, feature models, architectures, frameworks, security assets, production plans etc. are not created a new, but are adapted from the platform developed during domain engineering by binding variability. The effort required making each adaptation should be analyzed and those adaptations should be rejected which require an effort similar to that of developing the complete product anew.

Variability, during application engineering, is bound by providing specific values for component internal configuration parameters. Therefore, testing should be done to find out any defective configurations as well as to confirm that the correct variants are bounded.

Since there is no running application in the domain engineering, so only components or integrated data can be tested. The applications are available for testing only in application engineering phase.

Components in the core asset repository must have a defined variability mechanism so that these can be modified as per need for efficient use. Core asset repository should be regularly updated by adding new assets as product lines progress. The COTS present in or added to core asset repository must satisfy the return on investment (ROI) for the organization.

Security concerns should be incorporated in the SPLE and product line architecture from the initial stages as retrofitting security in the later stages might break the product line architecture. With regard to security, it must be noted that if the cost of security is higher than the data that needs to be protected, then the security is not worth the cost.

Lastly, it should be noted that if domain engineering is done right, the development effort in application engineering will be greatly reduced in comparison to a single system development.

6. CONCLUSION

In this paper, an improved framework for software product line development has been proposed. The framework illustrates a promising approach for the development of secure software applications using software product line engineering concepts. It also emphasizes on the incorporation of security requirements concerning a product line from the very initial stages i.e. in the domain engineering phase in a systematic way adhering to international security standards such as ISO/IEC 27001 standard, ISO/IEC 27002 standard, and ISO/IEC 15408-1:2009 standards. Further, the proposed framework is theoretically evaluated with selected state of art frameworks. In future, the proposed framework will be evaluated using empirical study and formal method techniques.

7. REFERENCES

[1] McIlroy, M.D., Buxton, J., Naur, P. and Randell, B., 1968, October. Mass-produced software components. In Proceedings of the 1st International Conference on Software Engineering, Garmisch Pattenkirchen, Germany (pp. 88-98). sn.

[2] Parnas, D.L., 1976. "On the Design and Development of Program Families", IEEE Transactions on Software Engineering, March.

[3] Clements, P., Northrop, L., 2001. "Software Product Lines: Practices and Patterns".Addisson-Wesley. 0-201-70332-7.

[4] DeBaud, J.M., Schmid, K, 1999. A systematic approach to derive the scope of software product lines". In: Proceedings of the 21st International Conference on Software Engineering, Los Angeles, California, United States.

[5] Khan, A.I., Alam, M.M., Shariq, M., May 2015. " A Perspective Study of Intelligent System for Component based Development". International Journal of Computer Applications 117(4):11-17.

[6] Khan, A.I., Khan, U.A., 2012. "An Improved Model for Component Based Software Development." Software Engineering 2, no. 4 (2012): 138-146.

[7] Khan, A.I., Alam, M.M., Jedaibi, W.A., January 2015. "Variability Management in Software Development using FeatureIDE: A Case Study." International Journal of Scientific & Engineering Research, Volume 6, Issue 1, 584 ISSN 2229-5518

[8] Deelstra, S., Sinnema, M., Bosch, J., 2004. "Experiences in software product families:problems and issues during product derivation".In: Proceedings of Third International Conference, SPLC2004, Boston, MA, USA. Springer.

[9] Kim, K. et al.,2006. "Asadal: A Tool System for Co-Development of Software and Test Environment based on Product Line Engineering". In: 28th International Conference on Software Engineering (ICSE), pp. 783–786.

[10] Dager, J. C., 2000. "Cummins' Experience in Developing a Software Product Line Architecture for Real-time Embedded Diesel Engine Controls," Patrick Donohoe (ed.) Proceedings SPLC1, Kluwer. Page: 23-46. ISBN: 0792379403.

[11] Hetrick, W.A., Krueger, C.W., Moore, J.G., 2006. "Incremental return on incremental investment: Engenio's transition to software product line practice". In: Companion to the 21st ACM SIGPLAN Conference on Object-oriented Programming Systems, Languages, and Applications, Portland, Oregon, USA.ACM.

[12] Pohl, K., Böckle, G., van der Linden, F., 2005. "Software Product Line Engineering". Springer, Berlin/Heidelberg/New York. 10 3-540-24372-0.

[13] Böckle, G., 2000." Model-based requirements engineering for product lines". In: Proceedings of the First Conference on Software Product Lines: Experience and Research Directions, Denver, Colorado, United States. Kluwer Academic Publishers.

[14] Kim, J., Park, S., Sugumaran, V. ,2008. "DRAMA: A framework for domain requirements analysis and modeling architectures in software product lines". Journal of Systems and Software, 81(1), 37-55.

[15] Tanhaei, M., Moaven, S., Habibi, J., 2010. "Toward an architecture-based method for selecting composer components to make software product line". Proceedings of the 2010 Seventh International Conference on Information Technology: New Generations (ITNG) (pp. 1233-1236).

[16] Mellado, D., Fernández, M. E., Piattini, M., 2010. "Security requirements engineering framework for software product lines". Information and Software Technology, 52(10), 1094-1117.

[17] Linden, F., Schmid, K., Rommes, E. , 2007." Software Product Lines in Action". Springer, Berlin.

[18] Guendouz, A., Bennouar, D., 2014. "Component-Based Specification of Software Product Line Architecture". In International Conference on Advanced Aspects of Software Engineering (pp. 2–4).

[19] Alam, M.M, Khan, A.I., 2013. "Risk-based testing techniques: a perspective study." International Journal of Computer Applications 65, no. 1 .

[20] Md Alam, M.M, Khan, A.I., Zafar,A, 2016."A Comprehensive Study of Software Product Line Frameworks". International Journal of Computer Applications 151(3):11-17.

[21] PWalters, JDMcGregor, Arcade Game Maker Pedagogical Product Line: Scope, Software Engineering Institute ,Carnegie Mellon University, Retrieved from http://www.sei.cmu.edu/productlines/ppl/product_line_overview.html

8. AUTHOR PROFILE

Mr. Md Mottahir Alam: is a Ph.D. scholar in the Computer Science & Engineering in Singhania University, India. He has six years of experience as Software Engineer (quality) for leading software multinationals, where he worked on projects for companies like Pearson and Reader's digest.

He is ISTQB certified Software Tester. He has received his Bachelor's degree in Electronics & Communication and Masters in Nanotechnology from Jamia Millia Islamia University, New Delhi, India.

Mr. Alam research interest includes Software Engineering esp. Software Product Line Engineering, Software Reusability, Component-based and Agent-based Software Engineering.

Asif Irshad Khan, Ph.D., is working as a faculty member in the department of Computer Science, FCIT, King Abdulaziz University, Jeddah, Saudi Arabia. He has thirteen years of experience as a professional academician and researcher.

Dr. Khan received Ph.D. in Computer Science and Engineering from Singhania University, Rajasthan, India, and Master & Bachelor degrees in Computer Science from the Aligarh Muslim University (A.M.U), Aligarh, India.

He has published several research articles in leading journals and conferences. He is a member of the editorial boards of international journals and his current research interest includes Software Engineering with a focus on Component Based and Software Product Line Engineering.

Dr. Aasim Zafar is working as an Associate Professor in the Computer Science Department, AMU, Aligarh. His current research interest includes e-learning, mobile learning, virtual learning environments and mobile ad hoc networks.

He received his Ph.D. degree in Computer Science from the Aligarh Muslim University, India. He has a number of research papers to his credits.

Dr. Zafar is a member of Internet Society (ISOC). He is a member of several editorial boards and a regular reviewer for reputed journals.