ISSN: 2394-3114
Vol-40-Issue-18-February -2020

A Diverse View on Feature-Oriented Programming in Software Product Line

U.Devi
Department of Computer Science & IT,
IIS University, Jaipur, India

N. Kesswani
Department of Computer Science,
Central University of Rajasthan, India

A. Sharma
Department of Computer Science & IT
IIS University, Jaipur, India

Abstract—The software product line (SPL) is a procedure of increasing products' set, in which variability is essential phenomenon taken into variable model. Fields & variables of SPL have been subject towide research over previousyears. Feature-oriented programming (FOP) is programming method for implementing SPLscreated on composition appliances known as refinements. In this paper, we have describedFOPwith software product line engineering (SPLE). Various researchers have worked on FOP. Different types of approaches for software product line also defined.

Keywords—Software Product Line,Software Product Line Engineering, Feature-oriented programming, Refactoring.

INTRODUCTION

SPL is the development of a set of engineering products that may be reused using general architecture & a predetermined plan. SPL's products may be embedded systems, software products, or software systems of some kind, such as digital services. SPL has become an important and popular way to improve quality, reuse support & efficiently achieve different product types. Variability management over years of research & training to become a central concern associated with SPLs [1].

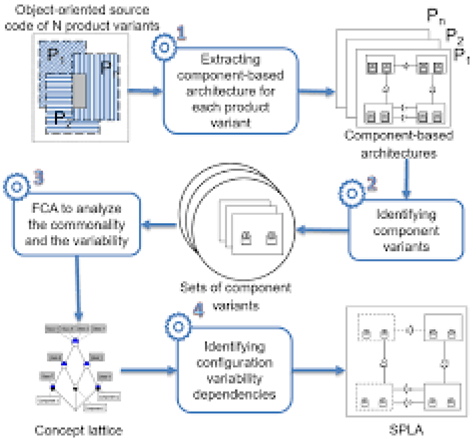
Object-oriented source code of N product variants
P1
Pn
P2
1
Extracting component-based architecture for each product variant
Pn
P2
P1
Component-based architectures
2
Identifying component variants
Sets of component variants
3
FCA to analyze the commonality and the variability
4
Identifying configuration variability dependencies
Concept lattice
SPLA

Fig.1. Software product line Architecture

SPL Development is a model towardsgenerating "a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way" [2].Based on recyclable platform, SPLDevelopment enables a large number of products to be customized to suit the needs of individual consumers with minimal development time & superior product quality [3]. Software discovery clarifies clear search links between all handicrafts in the software development process & applies them towardsthe software development process (SDP). [4].Although standardization has enabled the software industry to be significantly capable of software progress& to deliver affordable software for a wider market, it often does not address the needs and desires of small market segments or individual customers. SPLs combine standardization&mass productionby mass customization into software Engg. Software manufacturersmay create software productscreated on customer's needsfounded onset of refillable parts. Perception of aspects is critical towardsaccomplishing this automation'slevel, as it minimizes differences among customer needs & functionality product delivers. Aspects are central concept at every stage of product-line development. Writersproceeds approach of developer & specifically focus on improvement, maintenance, &execution of product line variables founded on user's feature selection. [5].

LITERATURE SURVEY

Inmaculada Ayala et al. [2019] Integrate targeted and SPL approaches to develop & guide the development of multi-agent structures intothe context of cyber-physical structures. We describe models'set (iStar 2.0 via target, CVL model via variable) &spontaneously work with agents on multiple heterogeneous devices, each by different configuration, defined by an algorithm for SPL process & development Propose process. We refer to this suggestion in context of domestic energy management system. Lastly, we verified scalability andpresentation of proposal bycasually constructed model. Consequences suggest that large iStar models of 10000 components may be handled in seconds with our approach [6].

Andras Kicsi et al. [2019] Suggest automated approaches to remove feature-to-program associates, with high-level aspectsdeliveredthrough domain experts. We associate call data by text comparisonamong code & high-level attributes for this determination. In addition, the discovery of communities between programs and feature engagements supports a deeper understanding of feature specificity. Because attributes originate by domain experts, community analysis exposesdifferencesamongspecialist's approach &interior code structure. We originate that groups with above half feature codes into specific communities fit well with high-level attributes. We report on two levels of testing and over 2,000 Magic 4GL programs into an industrial SPL adoption project [7].

Shatnawi et al. [2017] suggest amethod towardsthe rear engineer architecture of product's set versions. Our purpose is to recognizing variables & dependencies between architectural elementsalternatives. To analyze variability in our work relies on theoretical concept analysis. To confirmsuggestedmethod, we estimated2 families of open-source product variants; Health MonitorandMobile Media. Consequences of retrieved architectural variability & reliability accuracy and recall measurements were 81%, 91%, 67% and 100%, respectively. [8].

W. Fenske et al. [2017] suggest step-wise, semi-automated procedure towardstransfer cloned product variants towards facilitating SPL. Our procedure trusts on clone detection towardsrecognizinggeneral codes for several variants &new, & variant-protecting refactoring towardsremovinggeneral codes. We estimated our method on 5 cloned product variants & reduced code clone through 25%. In addition, we deliver qualitative statistics on possible limits & probabilities to remove more redundant code. We say that our methodmayefficiently reduce synergistic determinationrelated to clone-and-development &decrease long-term costs of preservation& development. [9].

J. Ghofrani et al. [2017] use consequences of certainconvolutional neural networks (CNNs) via code summation towards detect code clones. We utilizecreated description via2 code snippets like metric to measure similarity among them. We suggest a vector comparison measure via calculating the comparison index among these dimensions, which can determine which code snippets are cloned. [10].

R. V. Patil et al. [2015] Shows importance of finding equality. The similarity is found by operator or function overloading. Because it is an indispensable feature of noble object-oriented language. It discourses main methods that save time into retrieving & comparing data by extracting & configuring mining code by code document. The proposed system eliminates attempts to line-up code lines among 2 files following traditional algo. It describes reduction method & code complexity based analysis, increasing likelihood of success. The conclusion is that no single scheme describes procedures by detecting all types of clones. Weexisting multi-modal learning techniques to finding different kinds of code clones, which are occupied as problem statements into this task. [11].

H. Eyal-Salman et al. [2013]Suggests new method towardsincrease performance of IR approaches while useful towards software variants' group. The originality of our method is two-fold. On one hand, this uses what software variants (SV) have intogeneral& how they vary towardsincrease accuracy of IR consequences. Conversely, towards increase no. of relevant recovered links, it decreasesabstraction gap among aspects& source code withpresenting an intermediary level known as code-subject right of left. We applied our method towardsgathering7 variants of the large-scale systemswith ArgoUML-SPL modeling device. Trialoutcomespresented that our method transfers traditional application of IR approaches&latest&important work on topic [12].

A. Hemel and R. Koschke [2012] To explain this venture, CEWG examined us towards finding out how much up-and-upstream code may be originated into industrial products, & to what extent and to what fragment of kernel. We utilized clone detection methods extensively to relateseveral Linux versions with their vendor-specific alternatives. We originateseveralvariations that were not ported back. Certain changes have also been found in Linux subsystem, where neither we nor Linux Foundation suppose this. We originate improvements in conventional kernels that were not combined into vendor code. Complete, our examinationdelivers sufficient indication towards support requirementvia an LTSI &wellcooperation between Linux developers in mainstream and vendor variants. [13].

ISSN: 2394-3114
Vol-40-Issue-18-February -2020

Y. Jun et al. [2011] Suggest the use of an FOP method to differentiate CC into system / TLM (Transaction Level Model) modeling & to associate it withthe object-oriented approach. Evaluationdisplaysbenefits of our facility-based approach. The problem is important principle of SE. Given that TLM/ SystemC models are actually software programs that mimic nature of hardware, this principle ought to be surveyed intoTLM/ SystemC modeling & has an application in Separation of Communication & Computation (CC Separation). But, most of the remainingCSC/ TLM models do not differentiate into CCs, so changing their computation models or communication protocols leads to unnecessary development efforts. [14].

J. Ye et al. [2010]Refactor SoCLIB recommends using FOP method that reusable above problem and solve performance of each TLM-DT model. Though this paper may seem particular to SoCLib, concept of transaction-level modeling by FOP method is common. The transaction level is not an abstract level. Giving towards OSCI TLM-2.0 Language Reference Manual, this is moredistributed inApproximate Timeout (LT), Loaded Timed (AT), &Untimed (UT). Various use cases provide different sub-level service. But for each component, SoCLib deliverssolitary one kind of TLM, that is, TLM-DT. DT means the delivery time, & TLM-DT model may be comparedby LT model. Therefore this is major task to discover good way to improve two additionalkinds of TLMs viaeverymodule. This is unwise towardsimproving them by scratch as they share similar functionality bythe TLM-DTT model and can be reused in their implementation [15].

THE REFERENCE SOFTWARE PRODUCT LINE TESTING PROCESS

SPL test has a W-shaped life cycle [16] called extended V-model, which is composed of two overlapping V-models, as shown in Fig. 2. The arrows pointing by left to right into Fig. 2 show that domain test property is utilized as input towards application test. Test properties, for example test scenarios, test plans, &test cases, should be built into respective engineering stages. To identify some test properties, test engineers initiate system testing intoapplication or domainnecessities engg.stage, application architecture design or domain integration testing, application identification or unit testing intothe domain.

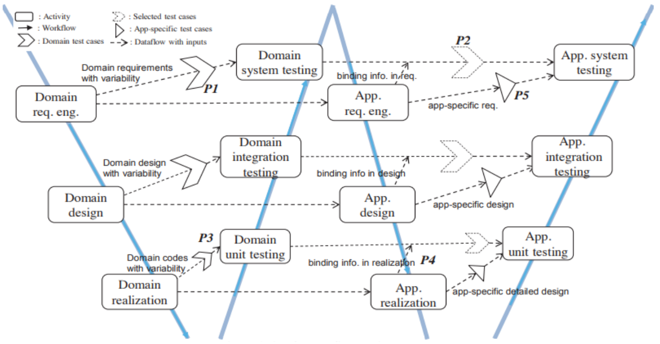

Fig. 2.A reference SPL testing process

Test propertiesmay be created &implemented through domain inspection or application verification. But, it is appropriate towards examining main property in domain testing therefore to application testing mayconcentrate on application-specific fragments that are not included in domain testing. Under normal circumstances, domain engineering is not focused on core asset development because it does not achieve the full range of products during domain engineering. So, in maximum cases, domain system testing maysimply be shown in restrictedmode[17].

In this view, SPL validation process in Figure 2 shows 2kinds of test cases for reference SPL testing, application test cases (test cases created in application Engg.)anddomain test cases (test cases built-in domain Engg.). Domain test cases must contain methods that may be effectively recycled intoapplication testing & address variables. In accumulation, test information should be determined via normality & variability for test cases [18].

I. Current Software Product Line Approaches

In this segment, several existing SPLmethods are provided as a synopsis. These methods are [19]:

A. FAST (Family-oriented abstraction, specification, and translation)

This is a feature-based model proposed by Weiss et.al [20]. It helps in applying product-line principles to software engineering process. It can be used in cases where a range of products are developed which have major share of common artifacts among themselves. These common features can be common behavior, common interfaces, or common code.

B. FODA

FODA [21](Feature-Oriented Domain Analysis) has been proposed through Kang to identify & model features. It is based on a domain analysis technique in which distinct features within a product line are identified. These features combined together to define the domain of the product family. This approach is followed because variability purpose mechanisms that are specifiedinmodules are producedwithintends of domain-specific language which is data-intensive allowance of textual version of feature diagrams.

C. FORM (Feature-Oriented Reuse Method for product line software engineering)

FORM (feature-oriented reuse method) is a feature-oriented technique of evaluating domain aspects [22] and then using these features to provide software feature line architecture. In other words, "form" is a systematic way of capturing the changes of common features & applications into domain & focusing on "features" to utilize analytical outcomes to improve domain architecture &modules. FORM method is useful for relating domain analysis outcomes to refillable&adjustable domain modules. There are definite guidelines that work for this.

D. RSEB

RSEB(Rise-Drive Software Engineering Business) [23] is a use-case driven reuse procedurecreated on UML notation. It is a repetitive, use-case-focused method that facilitates development & reuse of reusable object-oriented software. The main focus of this process is on use cases. Under this process, we first describe requirements of product line domain bysupporttoutilizing cases. Before, domain architecture and reusable artifacts are designed. Lastly, object models are created with the help of this architecture and artifacts which are mapped to the use cases [24].

E. FeatuRSEB

FeatuRSEB(Reusable Software Engineering Business) is introduced by bringing together FODA and RSEB methods. Two more processes from Foda namely Domain Engineering and Feature Modeling are used to initiate RSEB process. RSEB handles variables methodically in use cases, but no feature model was madeinthe procedure.

F. ConIPF

Its full form is Formation of Industrial Product Families & it is the European FP6 project [25]. This concept was put forward by Eriksson in whose words ConIPF is "a project which wants to integrate both the product line approach and the structure-oriented configuration 35 technologies".

G. PuLSE

Fraunhofer Institute Experimental Software Engineering (IESE) designed Product Line Software Engineering (PuLSE) in the late 1990s [26]. According to PuLSE method focus of SPLought to be on products before on organizational features. PuLSE is composed of 3main components viz. distributionstage, technical modules& support modules in that order.

H. KobrA

KobrA (Komponentbasierte Anwendungsentwicklung) technique has been establishedthrough Fraunhofer Institute Experimental Software Engineering (IESE). This is component-based SPL approach [27]. It is new approach in the fact that it is a grouping of reuse into small concepts into component-based methods&recycles intobig concepts in SPLapproach.

FEATURE-ORIENTED PROGRAMMING

Software variability [28] refers to combination of aspects that make up a product, &maybe denotedthrough feature model. Feature model [29] is formal paradigm for capturing and representing generalaspects& variations among products that make up a product. Figure 3 shows partial feature model of SPL Tankwise [30]. Certainaspects are precise to several platforms, for example,Handy&PC in this figure. Additionally, there are optional aspectsfor example map feature size (M_240, M_600, M_780 features).

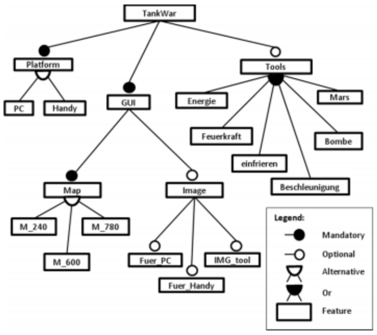

Fig.3. Partial Feature Model of TankWar

TowardsimproveSPL, we mayutilize various methods, such as explanatory & creative. Compositional methods, for example FOP, are implemented as features (physically distinct) code units. FOP is created on stepwise modifications. Stepwise modification is a model for developing complex programs to enhance detail from simple to simple [31]. Principal fractions &program increments are known as revisions & constants, correspondingly. Classes perform basic functions of system (constants) & these functions include attributes (modifications) in extensions.

FOP must not be restricted to the synthesis of source code. There are different other non-code artifacts (for example UML models, grammars, makefiles) that may be specified equational demonstrations, & be derived &madeas source code [32].

A. An Example of FOP

In itsegment we present feature-based programming founded on (partially) modeling stack bysucceedingaspects: [33]:

- **Stack:**provided thatpop andpush operations at stack.
- **Counter:**enhances local counter (utilizedvia stack's size)
- **Lock:**accumulation switch towardsdisallowingor allow alterations of an element (nowutilizedvia stack)
- **Bound:** which applies range check, utilizedvia stack items
- **Undo:**accumulation an undo function, which reestablishes state likethis was formerly last access towardsthe element

CONCLUSION

SPL testing has 2 distinct but close test engineering functions. Over the past decade, severalSPL testing approaches have been developed, & surveys have been conducted on them. Therefore, this paper identifies recentSPL testing approach withdescribing reference SPL testing process & what is important in SPL testing. SPL is used to develop families of similar software systems in the industry. The restructuring maytoo be done with specific refactoring towardsincrease internal quality of existing SPLs. FOP is a designedprocedure& tool via program synthesis. The goal is to clarify the features that the target program has to offer and integrate an efficient program that complements these features. FOP has been utilized towardsimproving product lines into wide variety of domains, comprising compilers via expandable Java languages, fire support simulators via US military, high-performance NW protocols,& program verification tools.

References

[1] Raatikainen, M., Tiihonen, J., & Männistö, T. (2018). Software Product Lines and Variability Modeling: A Tertiary Study. Journal of Systems and Software.

[2] P. Clements and L. Northrop, Software Product Lines: Practices and Patterns (AddisonWesley, 2002).

[3] Kim, J., Kang, S., & Lee, J. (2014). A Comparison of Software Product Line Traceability Approaches from End-to-End Traceability Perspectives. International Journal of Software Engineering and Knowledge Engineering, 24(04), 677–714.

[4] R. Wieringa, An introduction to requirements traceability, Technical Report, Faculty of Mathematics and Computer Science, Vrije Universiteit, Netherlands, 1995.

[5] Sven Apel, Don Batory, Christian Kstner and Gunter Saake, "Feature-Oriented Software Product Lines: Concepts and Implementation",Springer Publishing Company, Incorporated ©2013, book.

[6] Ayala, I., Amor, M., Horcas, J.-M., & Fuentes, L. (2019). A goal-driven software product line approach for evolving multi-agent systems in the Internet of Things. Knowledge-Based Systems, 104883.

[7] Kicsi, A., Csuvik, V., Vidács, L., Horváth, F., Beszédes, Á., Gyimóthy, T., & Kocsis, F. (2019). Feature Analysis using Information Retrieval, Community Detection and Structural Analysis Methods in Product Line Adoption. Journal of Systems and Software.

[8] Shatnawi, A., Seriai, A.-D., & Sahraoui, H. (2017). Recovering software product line architecture of a family of object-oriented product variants. Journal of Systems and Software, 131, 325–346.

[9] W. Fenske, J. Meinicke, S. Schulze, S. Schulze and G. Saake, "Variant-preserving refactorings for migrating cloned products to a product line," 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), Klagenfurt, 2017, pp. 316-326.

[10] J. Ghofrani, M. Mohseni and A. Bozorgmehr, "A conceptual framework for clone detection using machine learning," 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI), Tehran, 2017, pp. 0810-0817.

[11] Schulze, S., Thüm, T., Kuhlemann, M., and Saake, G., 2012. Variant-Preserving Refactoring in Feature-Oriented Software Product Lines. In proc. of 6th Workshop Variability Modeling of Software-Intensive System, p.73-81.

[12] Abilio, R., Vale, G., Figueiredo, E., & Costa, H. (2016). Metrics for feature-oriented programming. Proceedings of the 7th International Workshop on Emerging Trends in Software Metrics - WETSoM '16, pp. 36-42.

[13] Batory, D. (2003). A tutorial on feature oriented programming and product-lines. 25th International Conference on Software Engineering, 2003. Proceedings.

[14] Banerjee, M., Roy, S. R., & Kumar, C. (2012). Feature Oriented Programming: A step towards flexible composition of modular programming. 2012 1st International Conference on Recent Advances in Information Technology (RAIT).