International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-4, November 2019

Architecture Centric Software Development using Program Line Code

V. Sujay, M. Babu Reddy

Abstract: ***SoftwareArchitecture Centric Development Approach is regular and price outstanding in software program development technique in the software applications product lines. Traditionally used techniques in software improvement are very costly and unreliable in time period of high quality attribute and time to market products. If we are working in the same area then architecture centric development is very useful. In this technique we will reusable existing developed applications aspects for growing new software program alternatively of developing these software from scratch that are very time consuming and unreliable. To reduce the development time period we will reuse elements of from every stage of improvement to minimize the development time and provide better quality. Already developed, established and friendly components will be reused for development of new software. In this paper we will analysis Architecture Centric Software Development using Program Line Code. Which focus on which center of attention on high-quality attributes of software.***

Key words: ACSD&PLC, PLS, Program Generation, Meta Programming.

I. INTRODUCTION

Software Architecture provides the basic structure of a software system as well as the enforcement of system and structure growth. Almost every training includes basic programs, associations between them, and the features of both elements and interactions. The software system model is a symbol, equivalent to the architecture of a structure. It works as a development model of system and growth.

System Architecture consists of unique structural alternatives from possibilities in software design in relation to the creation of fundamental options that are expensive to move once implemented. For instance, there was a need for the system that supported the space shuttle launch vehicles to be very efficient and also reliable.

Architecture centric is the priority of the different stages of t he clarification: the creation, testing and structure lining of t he design are the top priorities of an explanations.The protot ype of the design validates the structure and provides as mo del for the future of the progress.Description of Software Ar chitecture is the primary object describing the selected techn ology.Other objects benefit of design:

(i).Recommendations for software and the use of patterns a nd phrases.
(ii).Structures of the item
(iii).Team structures

Software Architecture Centric Design Technique

The SACDT is a method of Designing Software Systemthat depends on software to provide its specified services in all r espects.It uses thearchitectural design to develop and improv e development continuously, enhance architectural drivers a nd minimize realistic consequences. Technical issues are rec ognized quickly and solved before the price or time line can be affected during final design and implementation [2].As important as technical specifications are non-functional requirements such as quality, modifiability, and usability, etc. The system design is highly influenced by these systematic features. Architects must continuously address this issue with the architectures they choose in order to support the resources and provide the features necessary by the collective of participants.

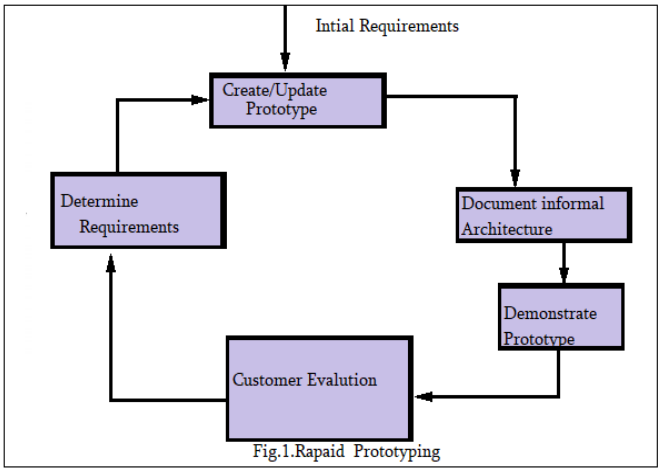

Fig.1.Rapaid Prototyping

II. BACKGROUND WORK

Design of software-centric product line integrates architectural design importance of Product line for software developmenta model that focuses framework recycling and massive specialization in software application system development [1].It works on product line structure (PLS) which contains all key elements specific for all product unit items and variation levels (e.g. an alternative element functionality) that identify product differences [2, 3]. Existing research and activities [4, 5, 6, 7, 8] has shown that based on PLS in SPLE is valuable as its architecture allows or prevents several of the quality attributes of a software system.

Manuscript published on November 30, 2019.
* Correspondence Author
V. Sujay*,Guest Faculty Of Computer Science And Engineering Sri Krishnadevaraya University, Anantapur, Andhra Pradesh, India.
Dr. M. Babu Reddy, Assistant Professor Of Computer Science Krishna University, Machilipatnam, Andhra Pradesh, India.

Generally extracted in SPLE, the consequent product objects (e.g. PLS, product line code) are collectively customized manually [10].

By correlation, the improvement of building plan driven item enables a PLS to be the essential concentration in item detailing, and item explicit code can be separated absolutely dependent on the customization consequence of PLS.

Now, many existing PLA driven product offering improvement approaches likewise center around full code age from the determined item explicit architecture[11,12] or utilize a program recovery framework to fundamentally change product offering code[13, 14]. Every single other system are conquering a few challenges thus can at same time.

In this article, we describe for a software product line a realistic architecture-centered method to product derivation. The system is concentrating on a procedure for coordinating PLS which broadens an example for code age and isolation with such a novel code arrangement method dependent on structure. It removes a PLS segment's product offering code into two unique classes: a code created from depiction of the product Architecture and the subtleties of the client characterized application. The delivered code can't be changed progressively and catches the utilization of compositional style-related contrasts.

The coding technique of design-based software is used in customer-defined software to support the automated integration of PLS variation levels. It identifies software segments related to various product functionality (e.g. methods, lines of code)[6]. The data in the annotations in specific is manually curated from PLS variation levels and cannot be misspelled. In contrast, a notation processor can upgrade its captions instantly.

III. ARCHITECTURE –CENTRIC PRODUCT DERIVATION

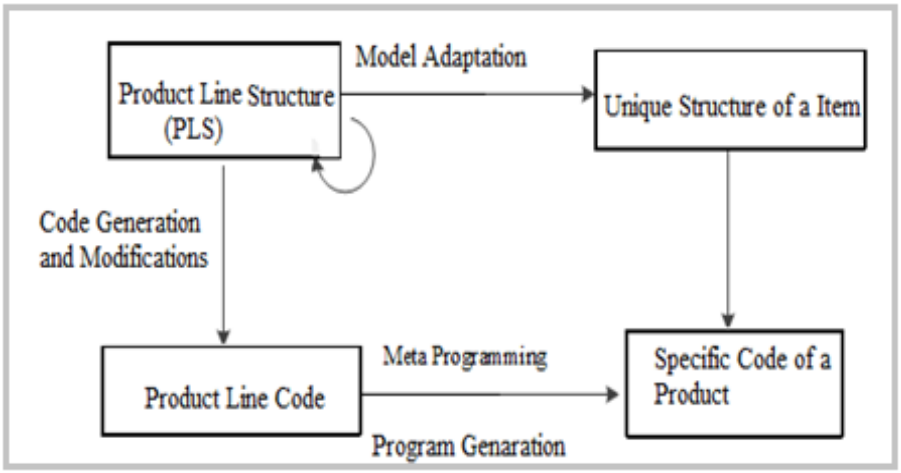

Figure.2.Two Established Architecture-Centred Product Conceptual Model Techniques

As Shown Above Structure, Two product derivation techniques used by several existing software architecture-centric SPLE StrategiesWhich actually begin from PLS Functionality activity where decisions are taken automatically for an item instance on a variation levels in the PLS.

As shown in figure 2 first receive PSA based on a few techniques of prototyping development.In primarily, the consequentialstructural design corresponds to a particularexecutionstage chosen used for the artifact. After that, it uses a code generator to generate entire Specific Code of a Product-from unique structure of item. A primary enhancement of such a method is how it improves its importance of structural design as structural design is used as main method in a combined effect of customization and framework. A challenge likely to face is the generation of massive script. Software design includes a software scheme's main design decision and is usually not necessary to obtain comprehensive application.

Product line code[PLC]still resides once the mechanism of f abric derivation starts. It involves the development of variati on in the line of products using general scripting languages. The mathematical notation method generates improvements to a Product Line Code due to changes provided to a PLS in PLS specification. The modifications derived may be configuration details or software development guidelines, based on the design technique used. And use a model dynamic languages technique or a software translation machine; they are applied to a Product Script. A main benefit of this solution is that it can support variations in

tasks throughout the product line and even performance characteristics.

In terms of views, it is not possible to resolve such similarities by providing software alone. A main issue of the approach is how it generates Item-Specific Code purely from Product Line Code specification and potentially reconnects Product-Specific Code from both PLS and Product-Specific Structure.

For Example, modifying the Product Code or the Product Specific Code effectively is complicated.

Ultimately, there is a deficiency for a software-centric method to product notation that can both apply centralized architecture and allow adaptive item-specific code and architectural design notation. We conclude this is related to how the existing approaches adopt PLS [15]. The very first technique above is based on prototype transition and code generation to gain levels of variation in the PLS.

Moreover, it can address varieties involved with platform. The second model is based on general-purpose software techniques to make changes in the PLS. According to the accessibility gap between both, it is difficult to relate the software to the difference areas in a PLS. Neither the script of the product or the software of the product can be revised effectively from PLS. It can minimize the scalability of the product code that's been established. For instance, this is observed in reality for the same element, there are sometimes different versions of the application to satisfy deviation goals characterized in the PLS

IV. APPROACH/METHODOLOGY

This section introduces a new Software Architecture centric method for the automatic formulation of product lines in a step of software applications. Except the existing solutions.
It permits the technology importance of a product providing specific use and implementation of a product both the product delivering software as well as the product specific code can only be rested by software level once the qualitative structure (e.g. PLs and product indirect prototype) changes. The technique may improve product outlines in phases and capabilities. The methodology PLS is shown using XADL, a depiction language based on XML.
The product derivation process which we have established is effective process. The product line code shown in figure is introduced use a system which integrates a sequence of code generation and segregation by a notation strategy based on architecture. The following classifications specific bring them. In other words, a PLS component's product line code was partitioned into 3 parts: a group including PLS code:
Product Specific Script is done effectively from subsequent Product-Specific Structure received from PLS, in view of the fact that vast bulk of the line of code is acquired from the regeneration by a user formed Product Line Code explanation processor. The actual product exemption operation provides three separately identified phases in this paper. Almost every step is clarified below.
1. Customization of PLS: It's the only step in derivation cycle which may require Additional interaction. Based on the feature specification, the developer defines the functionality to use or overcome all of the difference issuesin the PLS.For system support a range and consistency of a set of features and the detection of variety factors can be simplified.
2. Product derivation specific technology as well as the customer marked the specific code of the product. It development contains two free operations based on the preparations.
a. Specific innovation of a product. It suspends selected parts of the architecture (e.g. Design Pruning) and defines the specific structure of a product.
b. Object described by the user extracting specific script. A description processor is used to customize a Product Line Code defined by the consumer. It expels all the remarks shown and sections of the code which are not listed for the related examples. The effect is the product-specific code little defined by the user.
3. *Product-specific code [PCC]* generation. Suddenly, a code generator is used to produce product-specific software code derived from Step 2 instantly. The *Product-Specific Code* [PSC] can also be generated during process. The software generated code derived in Step 2 constitutes a consumer's final application.

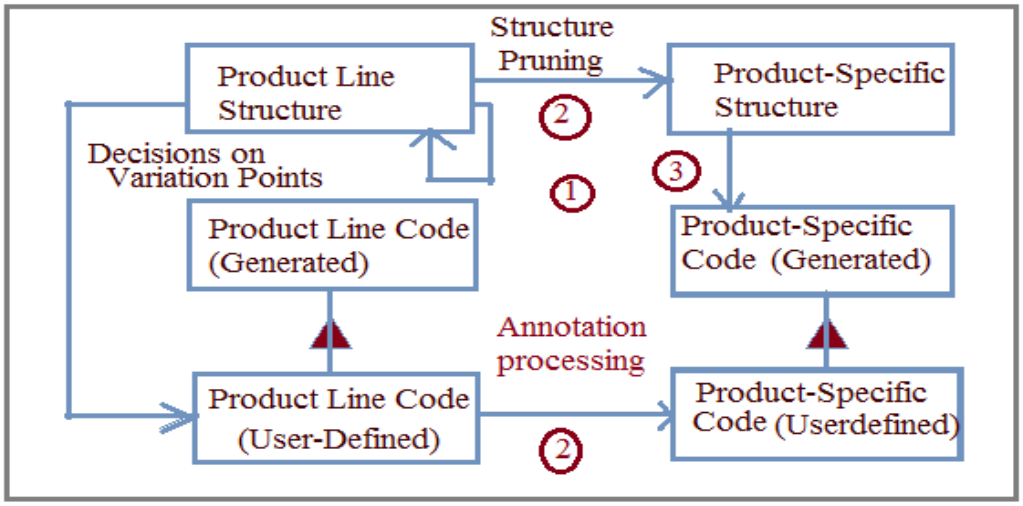

Figure3.A Conceptual Product Derivation Architecture-Centered Approach In Product Line Growth.

An example of a textbased chat application using the PLS el ement approach Part Storage at the top left has two solid fra meworks for the primary function and two scratched connec tors for an extra functionality. Interfaces also provided to refer to the inside element, storing the functions (e.g. rcvMsg) that are implemented in the component. The outward-pointing interfaces are allowed architectures and include the operations (e.g. fwdMsg) that the component uses and other components are implementing. The left-bottom is the specification of the part that contains a provided class
An interface derived and a user-defined class equivalent to the Figure 3 illustration. Server Arch contains archived code segments for the related aspects of the architecture. Paths 02 and 03 are dual parameters describing their specific applications required simultaneously, while lines 10-15 are the activities included in two applications supported.

Note that processes are performed by redirecting the offer to the Server Imp class described by client. Server Arch's referenced software segments show a code relevant to the default deployment model.

The right-top component server is a derived part of the architecture. It does not include the feature of the chat record and is implemented on a framework. The right-bottom always displays the relevant code generated. Class Server Arch and IServer are updated hourly from the derived factor of the architectural design. Notice that the code pieces for the removed code are not used.

Additionally, Server Arch contains various platform codes, from the product line software code. Server Imp is generated from the user-defined product code analysis and is changed accordingly.

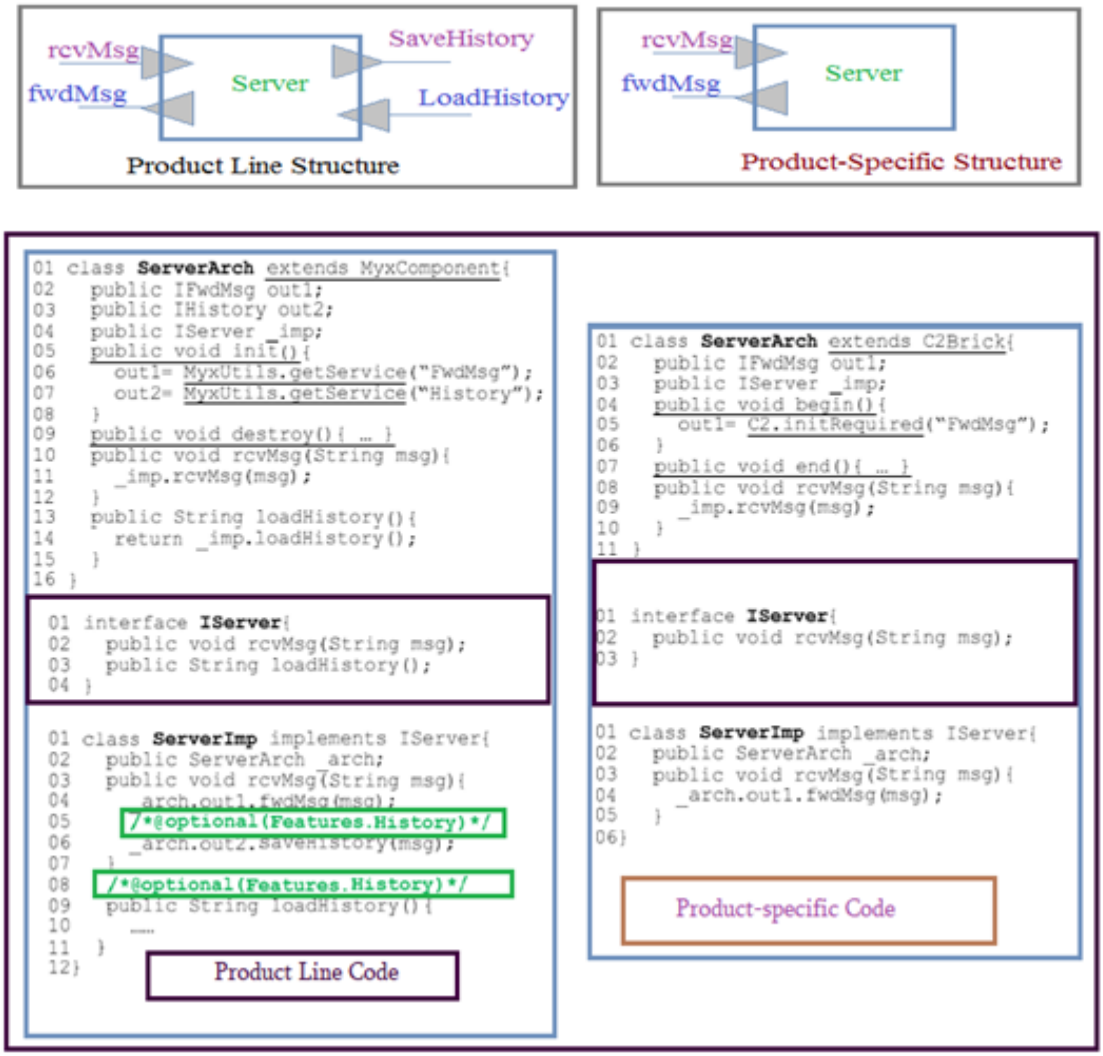

Figure.4.An Example of Product Derivation

V. CONCLUSION

In this paper, the mainly focusing on reusable applications and Established, friendly components will be reused for development of new software. We will analysis Architecture Centric Software Development using Program Line Code. Which focus on which center of attention on high-quality attributes of software. We also designed a method that includes a system for enforcing PLS and a method to dynamically derive product-specific structure and PLS code. It can support systems and features that involve both code generation and automated development disparities in a product line. We plan to further measure the method in a scope of the growth of a real product line in the long term. In future *Architecture Centric Software Development Using Design Pattern Techniques*

REFERENCES

1. Pohl, Klaus, Günter Böckle, and Frank J. van Der Linden. Software product line engineering: foundations, principles and techniques. Springer Science & Business Media, 2005.
2. Czarnecki, Krzysztof, and Michał Antkiewicz. "Mapping features to models: A template approach based on superimposed variants." In International conference on generative programming and component engineering, pp. 422-437. Springer, Berlin, Heidelberg, 2005.
3. Medvidovic, Nenad, and Richard N. Taylor. "Software architecture: foundations, theory, and practice." In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2, pp. 471-472. ACM, 2010.
4. Anquetil, Nicolas, Uirá Kulesza, Ralf Mitschke, Ana Moreira, Jean-Claude Royer, Andreas Rummler, and André Sousa. "A model-driven traceability framework for software product lines." Software & Systems Modeling 9, no. 4 (2010): 427-451.
5. Bosch, Jan. "Product-line architectures in industry: a case study." In Proceedings of the 1999 International Conference on Software Engineering (IEEE Cat. No. 99CB37002), pp. 544-554. IEEE, 1999.
6. Bosch, Jan. Design and use of software architectures: adopting and evolving a product-line approach. Pearson Education, 2000.

7. Groher, Iris, and Rainer Weinreich. "Integrating variability management and software architecture." In 2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, pp. 262-266. IEEE, 2012.
8. France, Robert, and Bernhard Rumpe. "Model-driven development of complex software: A research roadmap." In 2007 Future of Software Engineering, pp. 37-54. IEEE Computer Society, 2007.
9. Taylor, Richard N. "The role of architectural styles in successful software ecosystems." In Proceedings of the 17th International Software Product Line Conference, pp. 2-4. ACM, 2013.
10. Deelstra, Sybren, Marco Sinnema, and Jan Bosch. "Product derivation in software product families: a case study." Journal of Systems and Software 74, no. 2 (2005): 173-194.
11. Deelstra, Sybren, Marco Sinnema, Jilles Van Gurp, and Jan Bosch. "Model driven architecture as approach to manage variability in software product families." In Proc. of Model Driven Architecture: Foundations and Applications, pp. 109-114. 2003.
12. Kleppe, Anneke G., Jos Warmer, Jos B. Warmer, and Wim Bast. MDA explained: the model driven architecture: practice and promise. Addison-Wesley Professional, 2003.
13. Czarnecki, Krzysztof. "Overview of generative software development." In International Workshop on Unconventional Programming Paradigms, pp. 326-341. Springer, Berlin, Heidelberg, 2004.
14. Gray, Jeff, Jing Zhang, Yuehua Lin, Suman Roychoudhury, Hui Wu, Rajesh Sudarsan, Aniruddha Gokhale, Sandeep Neema, Feng Shi, and Ted Bapty. "Model-driven program transformation of a large avionics framework." In International Conference on Generative Programming and Component Engineering, pp. 361-378. Springer, Berlin, Heidelberg, 2004.
15. Gustafsson, Juha, Jukka Paakki, Lilli Nenonen, and A. Inkeri Verkamo. "Architecture-centric software evolution by software metrics and design patterns." In Proceedings of the Sixth European Conference on Software Maintenance and Reengineering, pp. 108-115. IEEE, 2002.

AUTHORS FROFILE

V. Sujay, Received His M.Tech. Degree From Srm University And Is Currently Pursuing Doctor Of Philosophy At Krishna University, India With Computer Science And Engineering Specialization. He Has Been Actively Involved In Teaching For The Past 7 Years And Now He Is Working As Guest Faculty Of Computer Science And Engineering At Sri Krishnadevaraya University, Anantapur, Andhra Pradesh, India. His Research Interests, Include Software Engineering, Database Management System, C Programming, Etc.,

Dr. M. Babu Reddy, Received His Master's Degree And Doctor Of Philosophy From Acharya Nagarjuna University, India With Computer Science And Engineering Specialization. He Has Been Actively Involved In Teaching And Research For The Past 18 Years And Now He Is Working As Assistant Professor Of Computer Science At Krishna University, Machilipatnam, Andhra Pradesh, India. His Research Interests, Include Machine Learning, Software Engineering, Algorithm Complexity Analysis, And Data Mining.