Enhancing Software Development through Software Product Line: Developing Product Family rather than Individual Products

Adekola Olubukola D. 1
1 Computer Science, Babcock University, Ilisan Remo, Ogun State, Nigeria,

Awodele Oludele 2
2 Computer Science, Babcock University, Ilisan Remo, Ogun State, Nigeria,

Abstract—Software Engineering is an art of designing software products for users’ consumption. This is an enduring knowledge area due to growing computational needs. Nobody succeed re-inventing the wheel. Re-usability is a key concept in software design. To an extent, the concept of software reuse has helped developers in meeting up with the market demands. Common reuse method includes using developed components, modules etc. to build new products. Yet, the traditional software engineering reuse patterns have not successfully addressed development challenges in terms of delivery time, cost and quality. This paper considers a new approach to reuse called Software Product Line Engineering (SPLE). This is described as “Industrial/Massive re-use”. While traditional software engineering focuses on developing individual products, software product line practice focuses on developing product family. To gain significant reduction in development time, reduced cost (both in development and products) and increase software quality, development is channeled towards SPLE.

Keywords-component; Software Engineering, Re-usability, developer, module, component, reuse, Software Product Line Engineering (SPLE), product family

I. INTRODUCTION

Software development process has recorded dramatic changes over the years consequent upon response to evolving users and market demands. Vital among many others are the need for low cost both of production and sales, timely delivery and higher quality. One of the most recent of these innovations is the adoption of Software Product Line Engineering in delivering software, hardware and engineering products. Product Line Engineering has really endured in the manufacturing industries. Bringing this into software production is quite an extended and elaborate way of thinking. The use of this in enhancing software development process is termed Software Product Line Engineering (SPLE). This could also be tagged industrializing software. Though more work is required to be put into the entire software engineering processes from requirement gathering to system delivery but the outcome is worth the extra mile.

The remarkable attribute of Software Product Line (SPL) that has afforded much acceptance is the industrial software re-use. This ushered in ample benefits in terms of shorter delivery period, quality and cost. Two important aspects in product line include identifying product commonality and deriving product variability. Product line combine development for re-use and development with re-use in a proactive manner. A more simplified explanation to this description is that designers employing this technique have reuse as first thing in mind [16] i.e. reuse is not an afterthought in the design process. It means the system is built for reuse from the word “go” where requirements are gathered till the product is finished. Therefore, a software product line model describes a set of products in the same domain (family of products) instead of a single software system.

In traditional software engineering, software architecture is evaluated with respect to the requirements of the individual product alone. Whereas a product line approach requires the software expert to consider requirements for a family of related / similar systems and the relationship between those requirements. There are three main areas of evolution process in PLE: Architecture, Components and Product.

From start, when products in the same domain are considered, elements shared by all the products in the line are realized and elements that may distinguish a product from another are also captured. The shared ones are regarded as the commonality described as core assets while the differing ones are the variants that differentiate one product from another called variable asset [16]. Core assets are those assets that form the basis for the software product lines. Suffice to say that SPLE introduces dimensions such as variability modeling and product derivation in engineering software products. The variability is where characteristics that distinguish a product from another are gathered and the product derivation is the complete process of constructing products from the software product line.

II. SOFTWARE PRODUCT LINES KEY DEVELOPMENT PROCESSES

A going concern is the need to develop multiple, similar software products rather than focusing on engineering a single individual product per time.
This would influence positively products which are to be consumed by the international market which must be adapted to different legal, cultural or language backgrounds / environments. It become quite herculean to develop or build systems one for each consumer using the traditional software engineering processes. This is because much development effort is expended and repeated irrespective of the quantity of components reused. Therefore, a concept that would maximize software reuse at a large scale level is called for. SPLE brings a dynamic approach to meeting this daring need [6]. This is a systematic way of producing families of software systems instead of creating a succession of completely individual products.

The SPL approach attempts to industrialize the software development process. Software Product Line Engineering is a proactive approach to software development that models what is common and what differs between product variants. The commonalities are the core assets of the family line and this forms the platform for strategic and planned re-use. The variability among the products is what defines a specific product in the family.

The development of products with differing characteristics based on a common platform is enabled by explicitly modeling the product line’s variability. The deferring characteristics of products with regard to a specific functionality are called variants. Variants are assigned to variation points. Variation points designate the possible decisions to be made in the derivation of a product. The variability of a product line is therefore defined as the sum of its variation points and variants. Developing a common platform and deriving customer-specific products gives rise to two major development processes namely [16]:

**Domain Engineering:** focus on identification and determination of the common features and the variability of a product line, the derivation of reference architecture and the realization of generic components and the associating quality assurance. Domain Engineering gives birth to development for reuse where core assets are factored out through domain analysis, domain design and domain implementation schemes.

**Application Engineering:** focus on the realization of the customized products by making use of product variability. This is characterized with products development. A differentiation of product is established by systematic binding of variation points with the predefined variants. Application engineering author development with reuse leading to the development of the final products through the following processes namely: application requirements, application design and application coding.

In the structure above, analyzing the domain produces a set of reference requirements reusable in defining an application requirements and integrating new requirements. The domain design defines a reference architecture which is used to develop and structure applications. And the domain implementation generates reusable components used in more or less seamless and time-efficient applications coding. A back and forth traceability is established between the reference requirements, the reference architecture and the reusable components to enhance the changes and updates management in the product line. At the application engineering level, a feedback/adaptation process can be used to revise or fine-tune the domain design and the domain implementation [14].

A number of approaches to implement SPLE in line with the above-mentioned development processes as well as some additional processes are in existence. Worthy of mention is Feature Oriented Domain Analysis (FODA). [4] FODA is effective in identifying commonality and variability among different product domain. Feature modeling is important to SPLE. Feature is an effective communication medium among stakeholders. It is different from other abstractions like functions, objects, etc. in that they are used to specify internal details of a system. While structured methods specify internal details of a system using functions which represent procedural abstractions, object-oriented methods specify the structures and behaviours of a system using objects. A feature is the key distinctive characteristic of a product.

Other approaches used in generating members of a product line include: the Family Oriented Abstraction, Specification
and Translation (FAST) process – introduces the idea of representing constraints among variability as edges in a graph and using different graph walking algorithms to generate members of a product line, Synthesis approach as well as FAST employs two stages namely: domain engineering and application engineering. Product Line Software Engineering (PuLSE), Sherlock and Odyssey-Domain Engineering [4].

Considering Nokia mobile product family as an example [8], an appreciable number of series of product is produced annually. Quite a number of languages and interfaces are supported courtesy of the fact that the product is sold in many parts of the countries of the world. Because a new product succeeds another quickly, there is need for quick delivery too. Also, consciousness for quality and cost must not be lost. Thus, employing product family engineering makes it possible to create software for the different products and use variability to customize the software in each different mobile phone. The first phase is the product management where the scope of the different mobile phones is defined. The second is the domain engineering, where requirements are defined for the family and for the individual type of phone. Here, the software platform is made and common assets are established. The third phase is the product engineering which focus on the individual types of phones.

The distinguishing characteristic of SPLE is the logical separation between the development of core assets (the platform) and the actual applications. The commonalities and variabilities are described in the Problem Space. This is mapped onto a Solution Space where constituent assets of the product line are described [6].

### III. SPLE REUSE PATTERNS AND OBJECTIVES

The traditional reuse practiced by most indigenous software industries is the type in which there is a repository where all finished products are stored. And when there is a need for a new design, this store is visited to fetch components, modules or algorithms that may be reusable. The act of searching, fetching and adapting this to the design at hand is rather cumbersome and tasking. Sometimes, building the new product from scratch may be preferable. Since SPLE follows the path of planned reuse, the platform produced greatly remove this possible bottleneck. The core assets product include major artifacts that are most challenging to develop namely: domain models, requirements, architecture, components, test cases, performance models and so on. This connotes a massive kind of reuse rather than just some code chunks. Since reuse is the subject for embarking on any SPLE development process, the following issues are naturally addressed by the final outcome (product family):

- Would we gain time to deliver on the overall (though much development effort is required in the course of this process)?
- Software quality as a result of less standardized component.
- Producing software as individual product is less proactive, less intuitive, boring and burdensome
- Code reusability advantage being under-utilized in traditional approaches
- Software development process need being industrialized to make software design a less tedious task
- Is there an approach to remove the question: who and what is responsible for product delay?

Over and above, recent practices in the development of service-oriented systems showcase that tasks such as systematic reuse and appropriate composition of services are not just tedious but also contain high propensity for errors. The application of Software Product Line Engineering approach appears to have a pretty exciting panacea to these protruding issues. The strength to leverage upon is the promotion of systematic reuse in the development of systems with similar requirements exhibited in SPLE.

From online literatures and findings, one can highlight that, as this new concept continues to mesmerize and gain audience attention, quite a number of companies have made the transition to software product line practice in order to keep pace with growing business demand for their products [19].
IV. SPLE REQUIREMENTS ENGINEERING

Requirements in software development “are descriptions of how the system should behave”[18]. The phase is simply to answer the question: what are we to build? But not how? The purpose is to understand the problem at hand. A problem misunderstood will inevitably get a wrong or different solution if at all a solution is provided. Most system failures are traceable to miss-spelt requirements. Even though the user knows what he wants, there might be pretty much difficulties in relating it to a system designer or analyst/expert. It was succinctly put that mostly, the users know what they want but they cannot explain their requirements effectively [1].

Speaking from perspective of the subject matter, Requirements Engineering processes and techniques are used to identify and characterize product line requirements and potential member products considering their commonalities and variability. Notably, activities involved include product line requirements modeling, specification, verification and management. At this juncture, it must be reiterated that the product line development process is quite complex and extensive, hence, requirements engineering is much more crucial for a software product line practice than for single product development. A whole lot of effort is required to be able to factor out requirements for a family of products or a quantifiable number of similar products. But, most of the existing software product line practices do not completely address the requirement engineering process exclusively.

Still, the dominant challenge remains that the requirement engineering process cannot be boasted about as far as capturing right requirement across a product line. Issues such as documenting wrong requirement, misinterpreting them, omission, careless commission errors or not having current ones poses a lot of hard times to system designer attempting to implement this approach. Product line requirements uniquely span several products. Therefore, only requirements specific to a product are regarded as a set of deltas relative to the common requirements in the product line (product domain). This consequently yield a lot of performance gain at requirement management level. This benefit the entire phases of requirement engineering activities which include requirements elicitation, requirement analysis and negotiation, requirement specification, requirement validation and requirement management [18].

V. SPLE GAINS

Advantages gained cover the stakeholders involved or all others that have something to do with the software design. Much celebrated is the consumer. There is hope of getting the product in good time especially when the particular product family is already in place. “Manufacturing” another product relating to the product line should be swift. The release is also guaranteed to be stable, standardized and of high quality. Although this approach might be tedious or demanding at first, there might be the risk of abandoning the project or need for staff training or restructuring, the eventual result is worthwhile. Various system builders seem to benefit at their respective posts since the SPLE reuse pattern cut across all phases.

VI. CONCLUSION AND FUTURE REMARK

Product Line Engineering (PLE) is one among most recent innovations in the delivery of software, hardware and engineering products. The driving forces among many others are the need for low cost both of production and sales, timely delivery and higher quality. Evidently, it means a win-win synergy between the end-user and the designer. SPLE is a systematic way of producing families of software systems instead of creating a succession of completely individual products. This offers industrial software re-use far beyond code reusability. The SPL approach attempts to industrialize the software development process. It forms the platform for strategic and planned re-use.

Most local/indigenous software industries are trying hard in meeting the ever-growing market demands as regards providing necessary automated solutions to local and foreign industrial problems. People desire software to drive their business faster but many a time, the software engineers and their growing industries seem not to meet up with delivery time. Many times, they lose these customers to on the shelf product which may not perfectly meet consumer’s need. A noticeable problem is the approach used by the software industries to build or design their products. They design per product and gather requirements per product. Then their reuse pattern is mostly to revisit component, modules, code etc. from an earlier built product. This slows down the entire software development process. SPLE approach is a paradigm shift that builds primarily for reuse. Then new products in this line will be released in good time, good quality and at low cost with the expectation of minimized maintenance. It is important to note that more future research is still required to be put into product derivation as far as SPLE is concerned.

REFERENCES


