International Journal of Engineering & Technology, 7 (4.29) (2018) 104-108

SPC

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper



A Novel Requirements Analysis Approach in SPL based on Collateral, KAOS and Feature Model

Fazal Qudus Khan[1]*, Shahrulniza Musa[1], Georgios Tsaramirsis[2], Sohail Khan[3]

[1]Universiti Kuala Lumpur, Malaysian Institute of Information Technology, Kuala Lumpur, Malaysia
[2]Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia
[3]Department of Computer, Deanship of Preparatory Year and Supporting Studies, Imam Abdulrahman Bin Faisal University, Dammam, Saudi Arabia
**Corresponding author E-mail:Fazal.qudus@s.unikl.edu.my*

Abstract

Software product lines (SPL) can speed up the production of software through the auto-generation of products and related subproducts, reducing the need for developer involvement. There are some proposals for defining the requirements for SPL, but they do not propose any method for identifying the eco-systems required for a successful life cycle of the generated product. In this research we propose the use of MEASUR Collateral Analysis for the purpose of identification of all the environmental systems and KAOS goal modeling linked with a feature model for ensuring that the features are aligned with the goals and objectives of the focal system. The proposed approach is demonstrated with a case study.

Keywords: *Feature modeling; KAOS; MEASUR Collateral Analysis; Software Product line engineering, Requirements Engineering.*

1. Introduction

This document Software Product line engineering is to utilize the shared assets or artifacts in order to support software reuse to the different variants of the Product lines. The competition in software industries forced to decrease the time to deliver the product to the market, increased large-scale production and to reduce the cost and time of adapting the changes in the requirements. These are some of the main challenges that SPL is handling in an efficient manner [1].

Software Product lines production follows feature oriented software development methodology, which starts from Domain Engineering part to know which features are part of product line and which are not, the domain design and specification. After the Domain engineering part, the Application Engineering part begins, which is about the product configuration and generation. Feature modeling is the core of software product line engineering and a de facto standard in modeling variability in SPL [2].

Software product lines do not have a de facto standard for requirements analysis and specification [3]. A number of attempts have been done in the past to connect goal-oriented approaches with software product lines and proved that it is possible. A number of these attempts are covered in the related work section. Goal analysis diagrams have a similar tree structure with the feature diagrams so connecting these diagrams is possible. The main benefit of this association is that features will be aligned with business goals and objectives. Additionally, the goals aid in setting the scope of the requirements analysis. The main difference between the proposed approach and older attempts is that we used the Keep All Objectives Satisfied (KAOS) goal-oriented model (explained in section 3.2) for creating the goal tree and we directly link the features with specific goals. KAOS is also capable of capturing conflicting goals and obstacles that may arise. Another difference in our approach is that we utilized Methods for Elicitating, Analyzing and Specifying Users' Requirements (MEASUR)'s Collateral Analysis (explained in section 3.1) for identifying all the systems of the product eco-system that are required for the successful development, launching, operation and termination of the product of a software product line. These increase the chances that no system is left-out as it applies a holistic approach to the development of the product.

According to our proposal, the collateral analysis of the product line should be conducted. In some cases, it may be possible to conduct collateral analysis for the domain of product lines specifying all the common components while in some other case the Collateral Analysis has to be done for each product. During this process all the common required eco-system products will be identified so their product line can generate them for all products of this product line, increasing the chances of a successful product life cycle without any additional development overheads. Once the collateral analysis has been completed, the KAOS model for the main product can be developed. Upon its completion, the feature diagram can be developed, but features must be derived and linked to a goal of the KAOS goal analysis tree. This will ensure that the features are aligned with the business goals and objectives. Later sections explain the methods of collateral analysis and KAOS in some detailed and demonstrate how the proposed approach can be applied with an appropriate case study.

The next section of this paper presents the related work. Section three explains in some detail the MEASURE Collateral Analysis, the KAOS method, and the feature diagram. Section four presents the proposed approach. Section five, demonstrates with a case study how the proposed approach can be applied. Section six concludes this work, presenting the limitations and possible future extensions.

Copyright © 2018 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

2. Related Work

There have been some attempts to link goal-driven requirements analysis with Software product lines. One of these attempts, presented in [4] proposed the use of goal analysis based approach for the identification of common features in software product lines as well as their configuration. The paper also identified a number of benefits of using such approaches such as stakeholder goal and requirements tracing and modification of the configuration based on stakeholder's preferences.

Researchers in [5] defined an approach for Requirements Engineering for Software product lines, based on an extended model of G2SPL [4]. A use case diagram was developed for the G2SPL that was used for aiding the generation of scenarios. The authors claimed that this approach supports continuous software evolution and because the use case is closely linked with the stakeholders, hence the requirements will satisfy them. The authors included no evidence or evaluation for their work and according to them, there were no tools supporting their proposed approach. Also linking the features to use case diagram does not necessarily guarantee that the stakeholder's needs will be satisfied because not all stakeholders are present in the use case diagram and for those who are present, it is not un-common that the analyst doesn't know what they really required.

Researchers in [6] presented the most common approaches for the analysis and specification elicitation for software product lines. The most common specification strategies are a product based specification, where the features of each individual product are specified one by one. In this approach, the features are linked to a specific product. An alternative approach is the feature based specification, where individual features are specified without links to any other features. Another approach is the family based specification. Products within the same family, share similar or the same features. Based on this, the specification can be written for all the features of the product line with variable parts for individual features. A more generic approach is the global specification. The global specification lists the specifications that all features of a software product line must comply with. An even more generic approach is the Domain-Independent Specification where features are specified for certain domains instead of a product line. This is used in this research in relation with MEASUR's Collateral Analysis as generic features can be developed for each domain based on the stages defined by collateral analysis for this domain. This will ensure that no ecosystem product is missed out and eliminate any development overheads as these ecosystems will be automatically deployed. For example, systems required for the development or backup etc., can be auto generated for the domain instead of a specific SPL.

The literature supports that there is an association between software product lines and goal analysis and that it is possible to use goal-driven requirements based approaches for feature specification. Apart from the analysis benefits, there is an additional benefit of using goal analysis with SPL. Researchers in [7] showed how feature models can be auto-generated by goal analysis models. The i* framework [7] was utilized in this research for modeling the features. The paper presented the mapping process in detail and included fragments of pseudo code for the complex parts proving that the mapping between goals and features is possible.

3. Definition of the Methods Used

This section presents the MEASUR's Collateral Analysis, the Keep All Objectives Satisfied (KAOS) goal analysis method and the feature model that is used in this research.

3.1. Collateral Analysis

The entire MEASUR's Collateral Analysis (MCA) was introduced by [8] and aims to be used as a list of all the possible subsystems and process categories that are required within the life cycle of an information system. Figure 1, illustrates the MCA method.

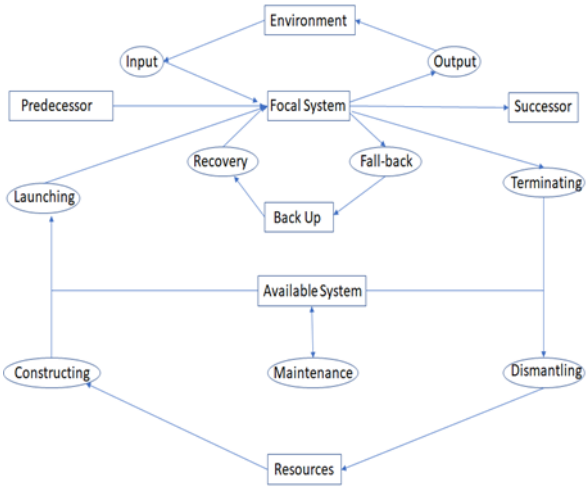


Fig. 1: Collateral Analysis method

The diagram in Figure 1 contains all the possible eco-systems for any information system [8]. The analysts must write down all the requirements about each subsystem and relevant processes for each node of the diagram in figure 1. This way it ensures that no subsystem or process is missing from the analysis. This is very important in the early stages where the duration of the project, cost and impact are estimated. The MCA requires from the analysts to think all the details about the construction, launching, operation and finally termination of the new information system. In more detail, as it can be seen in the diagram in figure 1, the method starts from the resources that are required for the construction of the new system which is referred to as the focal system. It then moves to any subsystems required for the construction of the new system. These may be either manufactured or bought. For example, maybe new computers are required and new software systems. It then moves to launching phase, which will usually take place while there is a parallel existing system running, that also has to be maintained until the new system comes into operation and the legacy system eventually is phased out. It is common to have both new and legacy systems running in parallel during the early days of the new system. MCA then moves to the new system. This system will require some input, e.g. data from the legacy system (predecessor) and must have a mechanism for generating output for its future successor system. For example this can be a script for extracting all its data. The system requires some input and returns some output to the environment. The new system must also have a backup subsystem/process that has fallback and recovery mechanisms. Finally, there should be a subsystem, process or plan for terminating the new system. Its termination will lead to dismantling that will release some resources (e.g. computers, staff, and space). These resources can be used in a new system. The use of MCA can aid software product lines as it can be used for setting the scope of any new system and increasing the chances that no eco-system or process is left out.

3.2. Keep All Objectives Satisfied (KAOS)

Keep All Objectives Satisfied also known as KAOS [9] is a requirement engineering approach developed by the University of Oregon and the University of Louvain (Belgium) in 1990 [9]. The key idea behind KAOS is that requirements should be justified by being linked with a business goal or objective and each of them should be associated with a responsible agent. KAOS offers an

integrated view of goal analysis, responsibility modelling, object modelling, and operation modelling in a single diagram [10]. In this research, we are only concerned with the goal analysis and responsibility part and not with the object or operation models, as in our proposal, these are replaced with a feature diagram (explained in the next section). Figure 2 shows the structure of a KAOS goal model with responsibilities.

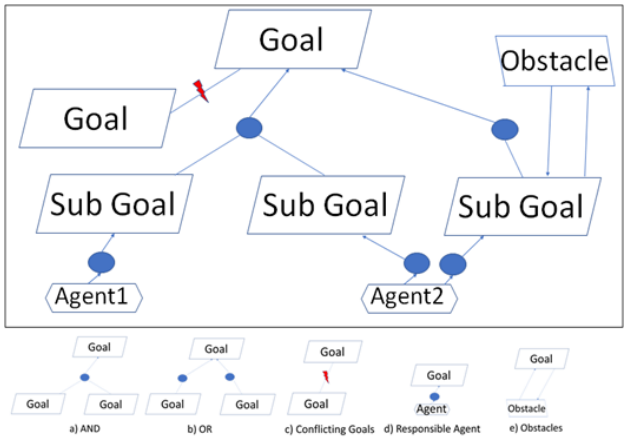


Fig. 2: KAOS Model

As it can be seen from Figure 2, KAOS model has one main goal and a number of subgoals that must be satisfied for the main goal to be satisfied. The association between the goal and the subgoals can be "AND" or "OR". subgoals connected with an "AND" association, must all be achieved for their parent goal to be achieved. If the subgoals are connected with the parent through an "OR" association, then if even one subgoal is achieved the parent goal will be achieved. Two goals may have a conflict with each other. In that case, if a goal is achieved the other goal will not be achieved. All end goals must be assigned to agents that are responsible for them. Agents can be physical or legal persons or roles. Last, some goals may be linked with identified obstacles. In this case, a management plan that will lead to the successful achievement of the goal must be developed.

3.3. Feature Model

Feature model [11] is a tree based structure used in Software Product Lines for illustrating the various features. The parent feature is called a compound feature because it consists of child features. Figure 3, presents a generic feature model. Level-2 Heading: A level-2 heading must be left-justified and numbered with an Arabic numeral followed by a period.

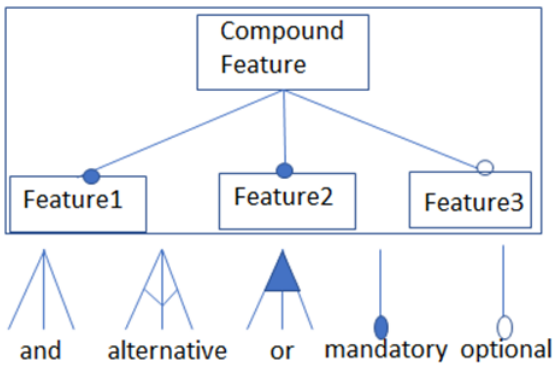


Fig. 3: Feature Model

As it can be seen from figure 3, there are three types of associations between the compound feature and its various child features. The "and" association implies that all features must be selected for the compound feature to work. The "alternative" (xor) association means that only one of the alternative features must be selected. The "or" association means that either of the features or even all of them can be selected. Each feature can be either mandatory or optional. The mandatory features are represented with a filled circle shape at the end of their association with their parent while the optional features are represented with a circle shape with no fill.

4. The Proposed Approach

In this proposal we recommend the use of collateral analysis as the first step. This is because it is important to identify all the system and subsystems that are required for the construction, launch, operation and finally termination of every target product. Such activities will increase the chances that nothing is missed out and since it is done via the software product line approach it does not add any major overhead to the development. Products of the same software product line are expected to have identical eco-systems and this process can be highly automated.

Once the Collateral Analysis is completed the next step is to conduct a KAOS analysis and generate a KAOS goal model with the goals of the business, any potential obstacles and the corresponding agents.

The last step is to generate the feature model from the goals. All high-level compound features must derive from low-level goals. Figure 4, presents an overview of the proposed approach.

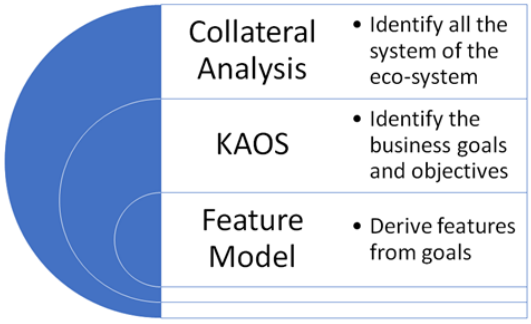


Fig. 4: The Proposed Approach

The proposed approach ensures that all the systems of the ecosystem have been studied hence minimizing the risk of problems at the later stages of the life cycle. The method also demands that the features are derived from business goals, ensuring that the product is fit for service. The next section demonstrates the proposed approach through a case study.

5. Case Study for EDR Product Line

In this case study, we cover the Electronic Display Room (EDR), as the software Product line for the Accreditation Board for Engineering and Technology (ABET) activities. EDR is a product line with many systems and subsystems covering the various areas of ABET accreditation process. EDR comes as an improved version of the KSO analytic System (KAS) used for a similar purpose. All the products were used at the Faculty of Computing and Information Technology at King Abdulaziz University, Jeddah, Saudi Arabia.

ABET accreditation is based on nine criteria which revolve around continuous improvement of the quality of students, faculty members, facilities and institutional support. The KSO Analytic system (KAS), which can be seen in figure 5, was a system that was used to generate an analytical report for the courses based on some inputs from the user for example, student's scores in different assessments. The EDR system served as its successor and it has many new features. We re-used some of the features of the KAS, while adding some new features based on the requirements (goals) of the accreditation unit which were to generate different kinds of reports. There were scheduled reports and some drill down reports

that were generated at the end of each semester for the courses of the college. The scheduled reports were for the faculty members' data and included information like, how many publications were

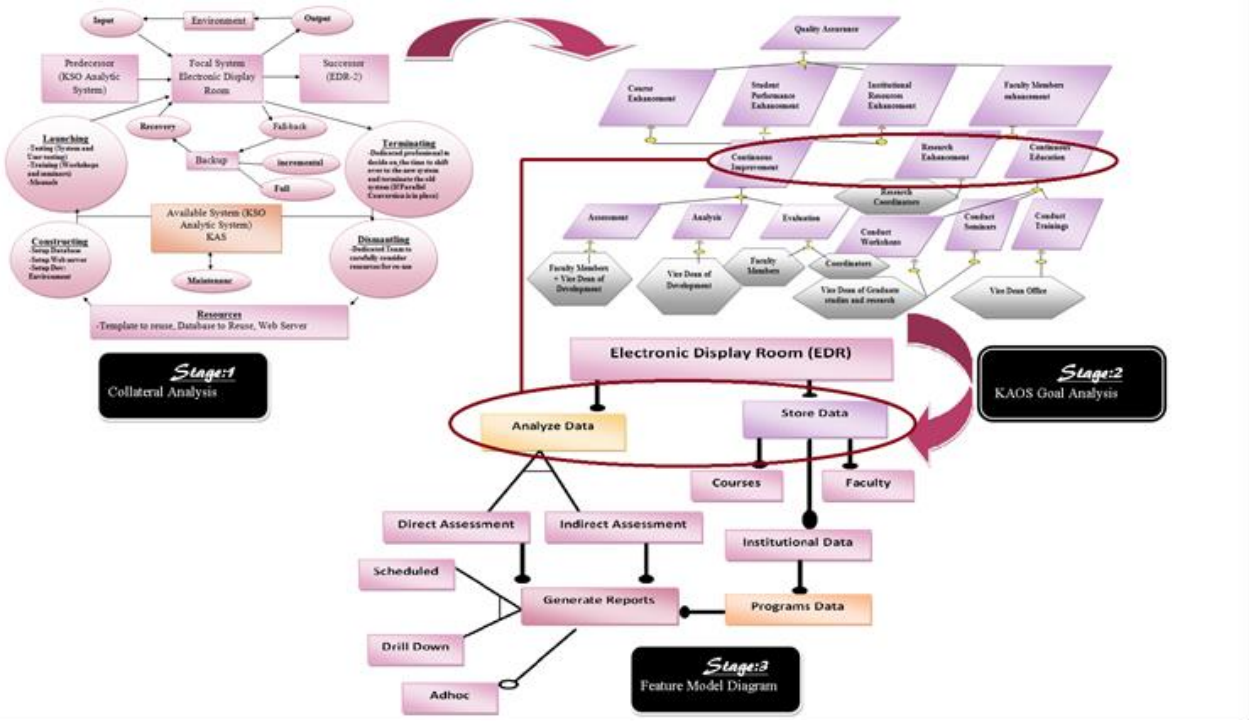


Fig. 5: The EDR Case Study

produced in a specific academic year and continuing education data such as training and workshop attended and so on. The drill-down reports were containing information for the whole department i.e. on what percentage the courses data were archived and which member of the faculty has not yet archived? It also provides details of how much percentage of archiving and evaluation reports are completed for a specific department. Figure 5, illustrate our methodology for the EDR system.

As it can be seen in the figure 5, the Collateral analysis was used to make sure that all the components necessary for the development of a new system are in place. Among the required features, was a development environment with a database and webserver for the implementation and testing of the features. Another required feature was a testing system for testing the system before launch. Additionally there was a requirement for Computer based training systems (CBTs) for training the staff on how to use the new system. An external backup system feature was also deployed prior to the launch, to support the operation of the system. Also, a data migration component was developed for transferring data from the old system to the new. The same component can be used in the future if there is a successor system.

The KAOS goal analysis diagram was used to identify the goals of the EDR system. Such goals included, course enhancement, student performance enhancement, resources enhancement and faculty members enhancement. The first three can be achieved, by continuous improvement while faculty members can be improved by research activities and continuous education. Apart from the goals, the KAOS diagram also identified the agents associated with each goal. These agents can be used as contact points for requirements clarification and if possible user testing.

The goals modeled through KAOS diagram are then linked with Features of the EDR system, which comprise of two main subsystems, One is for the analysis of course data against course learning outcomes and Student outcomes. The second subsystem is for archiving or storing data of the faculty members and courses along with institutional data. More features are in line to be added to the new product lines of the EDR such as conducting indirect assessment, analyzing and generating automated evaluation reports, and automated Self-study reports for the ABET accreditation purposes.

A number of benefits were observed by applying the proposed approach to this case study. All the features such as archiving / storing data and analyzing the data were aligned with the goals and objectives of the focal system such as continuous improvement, research enhancement and continuous education. Reduction in cost was noticed as a result of the reusability of components. The Software product was operational for two years facilitating all the needs of the set goals of the EDR. This was achieved because the method applied a holistic approach ensuring that no system was left-out.

6. Conclusion and Recommendations

This paper proposed a new approach for Software Product Line requirements analysis. The new approach was based on the merging of three approaches. The MEASUR Collateral Analysis, the KAOS approach and the Feature model.

Collateral analysis is conducted, in order to increase the chances that no support systems that may affect the life cycle of the product are left out. Then a KAOS goal analysis approach is used in order to identify the business goals that need to be meet by the systems and subsystems (features). A feature diagram where all features are derived and linked with goals from the KAOS model must be developed. This ensures that the features are aligned with the goals of the information system. The proposed approach was demonstrated through a case study for a system called EDR responsible for accommodating the requirements for ABET accreditation.

The main limitation of this work lies in the limited use of collateral analysis by the case study as well as the lack of tools for automating the process. In the future we will focus on the development of tools that will aid the features identification and selection.

Acknowledgement

This work would not have been possible without the support of EDR (AIMS system) which is the main system used for ABET accreditation required documentation.

References

[1] Apel, S., & Kästner, C. (2009). An overview of feature-oriented software development. *Journal of Object Technology*, *8*(5), 49-84.

[2] Khan, F. Q., Musa, S., Tsaramirsis, G., & Bakhsh, S. T. (2017). A study: selection of model metamodel and SPL tools for the verification of software product lines. International Journal of Information Technology, 9(4), 353-362.

[3] Sridhar Chimalakonda and Dan Hyung Lee. 2016. On the Evolution of Software and Systems Product Line Standards. SIGSOFT Softw. Eng. Notes 41, 3 (June 2016), 27-30. DOI: http://dx.doi.org/10.1145/2934240.2934248

[4] Silva, C., Borba, C., Castro, J.: A Goal Oriented Approach to Identify and Configure Feature Models for Software Product Lines. In: Proc. of the WER'11, Rio de Janeiro, Brazil (2011)

[5] Guedes, G & Silva, Carla & Castro, J. (2013). Goals and scenarios to software product lines: The GS2SPL approach. CEUR Workshop Proceedings. 1005. doi: 10.1109/APSEC.2004.56

[6] T. Thüm, S. Apel, C. Kästner, M. Kuhlemann, I. Schaefer, and G. Saake. Analysis Strategies for Software Product Lines. Technical Report FIN-004-2012, University of Magdeburg, 2012.

[7] Dongjin Yu, Zhenli Chen, Yifei Zhang, From Goal Models to Feature Models: A Rule-Based Approach for Software Product Lines, Published 2015 in 2015 Asia-Pacific Software Engineering Conference, DOI: 10.1109/APSEC.2015.22

[8] Kecheng Liu, Weizi Li, Organisational Semiotics for Business Informatics, Routledge, 2014, figure 4.2, DOI 10.4324/9780203550977

[9] A. van Lamsweerde, E. Letier. From Object Orientation to Goal Orientation: A Paradigm Shift for Requirements Engineering. Proc. Radical Innovations of Software and Systems Engineering, LNCS, 2003.

[10] KAOS Tutorial, Respect-IT, Objectiver, 18-10-2007, www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf

[11] Don Batory, Feature Models, Grammars, and Propositional Formulas, Software Product Lines 9th International Conference, SPLC 2005, Rennes, France, September 26-29, 2005. Proceedings