# VariaLBD: Approach for Modeling and Implementing Variability in the Databases Inherent to Software Product Lines

Nesrine Khalfallah[1*], Naoufel Kraiem[1,3], Sami Ouali[2]

[1] RIADI Laboratory, National School of Computer Science, Campus of Manouba, Manouba 2010, Tunisia.
[2] College of Applied Sciences, Ibri, Oman.
[3] Computer Science Department, College of Science, SQU, Oman.

* Corresponding author. Tel.: +21695740444; email: khalfallahnesrinee@gmail.com
Manuscript submitted March 1, 2019; accepted June 15, 2019.
doi: 10.17706/jsw.15.1.1-11

**Abstract:** In software product line engineering, a properly functioning of a software system requires effective collaboration and synchronization between the application code and the database. The main factor of the application's source code evolution is variability management. In fact, the software product lines are a method of software engineering that proves their effectiveness in variability management. Despite the database is an integral part of software systems, the software product lines are principally used for the executable code production. The effects on data management and in particular on database schemas still imperfectly documented and studied. However, they have many interests for the evolution of the whole software system. Consequently, the database evolution remains backward relative to that of the code. So, to ensure the harmony, the reliability and a smooth execution of the overall production process, this evolution must be parallel to that of the source code in the software product line. This is why it seems necessary to study this research line. Hence, in the context of this paper, the use of software product lines' techniques is proposed as a solution to solve this problem in order to take advantage of their potential for variability management. To ensure database evolution, this paper proposes an approach called VariaLDB based on the model driven engineering. Then, it presents an experimentation of VariaLBD on a case study. The experimentation materials developed especially for the VariaLBD test and an evaluation and a validation of the experimentation results will be presented in this paper.

**Key words:** Database schema, Feature, Model transformation, Software product line, Variability.

## 1. Introduction

Software systems have a major part that is the application code. Another important part which is present in many software systems is the database. A properly functioning of a software system requires effective collaboration between these two parts. In the past decades, software systems knew a remarkable evolution thanks to the continuous evolution of the code part. However, the database (BD) part is neglected and its evolution remains backward relative to that of the software. The main factor of the application's source code evolution is variability management. Indeed, variability [1] plays an important role in tailoring products to the specific user's needs, while common needs are present and reuse by all members of the products family. In fact, the software product lines (SPL) [2]-[4] are a software engineering method that proves their effectiveness in variability management. It is interested in a group of similar software systems

sharing a set of identical and different features. Despite the DB is an integral part of software systems, the SPLs are principally used for the executable code production. The effects on data management and in particular on DB schemas are still imperfectly documented and studied. However, they have many interests for the evolution of the whole software system. This is why it seems necessary to study this research line. Hence, in this paper context, the use of SPLs' techniques is proposed as a solution to solve this problem in order to take advantage of their potential for variability management. So, to ensure DB evolution, this paper inspired from SPL techniques to manage variability by considering all DBs set as a SPL and whose product is a particular DB variant. Thus, to ensure DB evolution, this paper proposes an approach called VariaLBD based on the model driven engineering (MDE).

In fact, we present VariaLBD and its details in Section 2. Then, in Section 3 we illustrate the VariaLBD experimentation on a case study "Smart phone product line". The experimentation materials developed especially for the VariaLBD test is presented in the same section too. Moreover, in Section 4 we evaluate the experimentation results and we validate VariaLBD experimentation. In addition, we quote some existing related work in Section 5. Finally, we conclude this paper with conclusions and perspectives.

## 2. VariaLBD Approach

VariaLBD [5], [6] is an approach to model and implement variability in DB schemas inherent to SPLs. Indeed, in the VariaLBD definition we are inspired by SPLs engineering techniques. So, the main idea of VariaLBD was to consider a set of DBs as a product line whose product is a particular DB with a variable schema. In fact, VariaLBD has as inputs a source class diagram and a feature model and as output a conceptual and variable DB schema inherent to SPL. VariaLBD treats the both levels of the SPL engineering. The first level is the domain engineering where VariaLBD works on a conceptual level to solve the variability modeling problem in the DBs. VariaLBD proposes a general meta-model [5], [6] for modeling variability in DBs inherent to every SPL. Then, we refine the modelization to manage the variability in a lower level. At this level, there is a passage from a source model to a target model that manages variability through a model transformation using the MDE. This transformation is performed through the definition of transformation rules to ensure the transition from a source class diagram to another target one. In addition, VariaLBD offers two different types of variability based on the location of variability in the diagram components, which are: *attribute variability and class variability*. So, for each type of them, we define specific transformation rules to ensure the variability modeling in the conceptual DB schema. Thus, VariaLBD propose three new stereotypes to manage variability in the conceptual DB schemas. These stereotypes classify the diagram's classes into three principal types. First, we propose the stereotype *<<normal>>* for real-world entities in target diagrams. Second, we propose the stereotype *<<type>>* for each class designed to contain variability in the target model such as optional or alternative or multiple choice attributes or classes. Indeed, classes stereotyped <<type>> generate *parameterization tables* at logical and physical levels which create and organize the variability in the physical DB schema. Third, we propose the stereotype *<<administrator>>* for creating a class that allows the configuration of the DB schema from the parameterization tables. Since this class produces after normalization a table that manages the different parameterization tables according to the features chosen by the user. So, the class stereotyped <<administrator>> must be created in every target class diagram because it has the conductor role by synchronizing between the different classes stereotyped <<type>>. Consequently, at this level the VariaLBD contribution is the generation of a conceptual and variable DB schema thanks to the use of these new proposed stereotypes to model the variability. Then, we obtain logic, relational, and variable DB schema after the normalization. The second level is the application engineering where configure the logic model generated in the first level and implement the variable and physical DB schema.

Thus, every change in the DB can be taken into consideration at the level of the conceptual DB schema in order to regenerate a new relational, variable, and evolutive schema. Therefore, VariaLBD indirectly supports the continuous DB evolution in the SPL by variability management. So, VariaLBD guarantees the DB schema evolution in SPL on the different conceptual, logical, and physical levels. In fact, VariaLBD manage variability from scratch in DB inherent to SPL because it begins the process by modeling the conceptual and variable DB schema. Thus, every change in the DB environment must be presented in advance in the conceptual DB schema to be considered later in the logical and physical schemas. An illustrative schema of the proposed approach VariaLBD is presented in Fig.1.

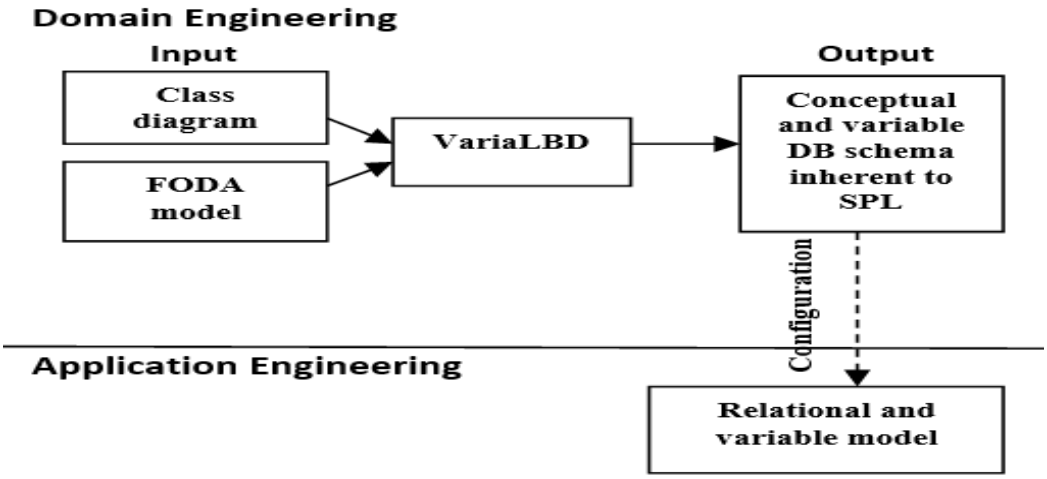


Fig. 1. Illustrative schema of the proposed approach VariaLBD [6].

## 3. Description of VariaLBD Experimentation

This experimentation is elaborated according to the experimental plans proposed in the literature [7]-[12] for approaches experimentation on confirmatory case studies.

### 3.1. Experimental Context

The case study proposed in this paper is a smart phone product line (SP). It's an industrial case study. It is used to test VariaLBD and evaluate its validity. This experimentation focuses on the modeling of variability in smart phone product line. Indeed, we must manage variability in the DB schemas inherent to this SPL during the production of the smart phones family. In this SPL, developers face a lot of DB configuration problems because of the lack of a variable schema that are automatically adapted to the change of customer needs. Hence, we proposed the use of VariaLBD as a solution.

### 3.2. Experimental Design

The experimental design contains the conceptual models used in this experimentation, which are: the class diagram modeling the studied DB of the case study smart phone product line and the FODA feature model of this DB. In our case, FODA's model enables identifying the variability in the various DB's constituents and specifying their variability type.

Indeed, these two models constitute the experimentation inputs. Fig. 2 and Fig. 3 show respectively the SP's class diagram and the SP's FODA model. In fact, these two models are the inputs of VariaLBD. The class diagram shown in Fig. 2 is the diagram that VariaLBD should model its variability.

### 3.3. Experimental Material

We proposed a website as a tool support [13] for the execution of VariaLBD which ensures her test on its case study. Indeed, this website is an open online space that enables users to create variable and relational DB schema inherent to any SPL. In fact, our tool enables automatic modeling of the source class diagram and the feature model FODA at the conceptual level. In addition, our tool transforms automatically the source class diagram to another target and variable one thanks to the implementation of the proposed transformation rules. Then, it normalizes automatically the passage from the conceptual level to the logical level.

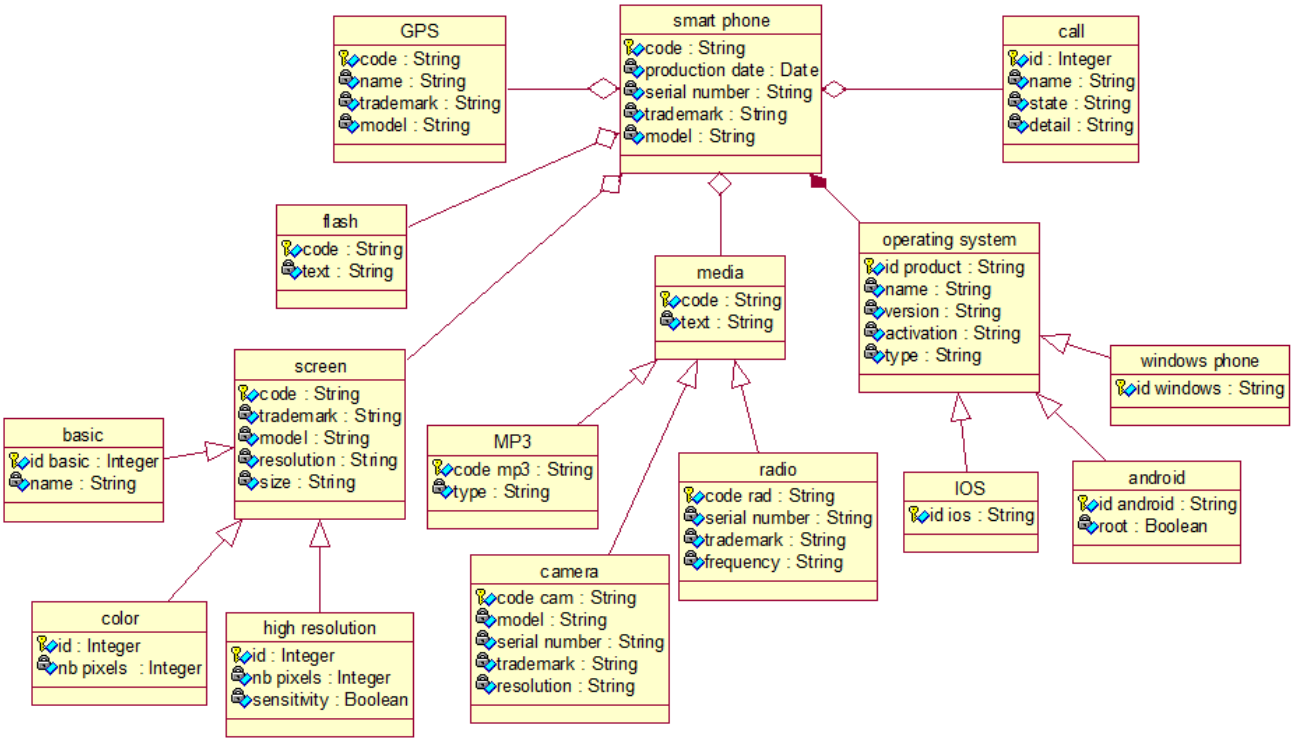


Fig. 2. SP's class diagram.

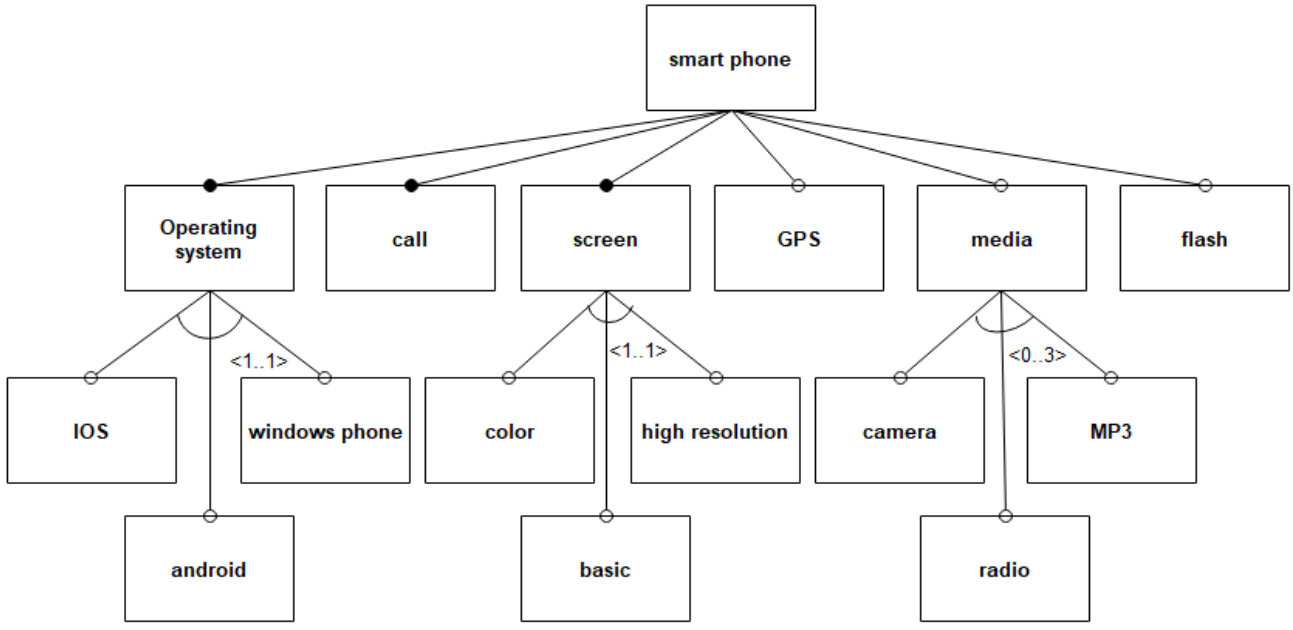


Fig. 3. SP's FODA model.

### 3.4. Preparation and Collection of Data

Data is collected from the real word according to an empirical study to create the class diagram and the FODA model. Then, after an analysis of these two diagrams, the variability is localized in the class diagram based on the FODA model. In our case study, the variability resides at the level of the classes. Indeed, the classes "operating system", "call", "screen", and "GPS" are mandatory classes. Whereas, the "GPS", "flash", "media", "IOS", "android", "windows phone", "color", "high resolution", "basic", "MP3", "camera", and "radio" are optional classes, their presence and their absences are the core of the variability in the studied DB of the SP product line. Thus, thanks to this location we can fix the type of transformation rules to be applied in this experimentation. Hence, they are the rules related to variability within classes.

### 3.5. Experimental Procedure and Execution

The execution phase is the core of this experimentation since it allows identifying the different experimental results and to evaluate the VariaLBD validity. So each task of the experimental procedure must be executed carefully to achieve the final product which is the variable DB schema. In addition, this phase is fully automated by the proposed tool support. The tasks in this procedure are explained below one by one.

#### 3.5.1. Data entry

The first task to do is entering the general information about the system's class diagram which will be modeled and all the necessary information about its entities, its attributes, its associations, and all their properties. After entering the class diagram, the user enters the variability data relative to this diagram. These data are extracted mainly from the FODA features model, like the type of each feature (optional or mandatory), the definition of alternative groups or/and multiple choice groups, and the definition of constraints between these features.

In this experimentation, the proposed tool support also offers the possibility of directly entering the feature model and its dependencies as an input model for VariaLBD.

#### 3.5.2. Transformation

This VariaLBD's task is automated thanks to the implementation of all necessary rules for transformation and for normalization in the proposed tool support. Transformation rules are applied to the class diagram created in the previous task to generate a new class diagram modeling the variability and representing the conceptual and variable DB schema. This new class diagram allows generating after normalization a relational and variable DB schema automatically.

### 3.6. Experimental Results

This experimentation has as a result the transformed class diagram which models variability within the conceptual DB schema of the SP product line. This class diagram is modeled using the proposed tool support. Then, the relational and variable DB schema of the SP product line is obtained after automatic normalization of the conceptual model. Indeed, this schema represents the final result and the VariaLBD contribution. Moreover, the FODA model of the SP product line is generated automatically from the class diagram entered by the user at the beginning.

## 4. Evaluation of the VariaLBD Experimentation

### 4.1. Qualitative Analysis of Experimental Results

To prove the validity and the reliability of VariaLBD, in this paper we analyze qualitatively the variable DB schema generated by VariaLBD. The qualitative analysis is purely theoretical. It is based on a theoretical and bibliographic study of the contribution characteristics of VariaLBD and of each existing approach. The studied approaches are the most relevant approaches in the literature [14]-[28]. Indeed, this study is

interested in extracting all data concerning the quality factors of conceptual models and DB schemas [29]-[31] for each existing approach. Also, for VariaLBD all the quality factors of conceptual models and DB schemas that may be useful for this analysis is identified throughout its definition. Moreover, this analysis consists on making a comparison between the DB schema produced by VariaLBD and the DB schema produced by the existing approaches. In fact, the comparison criteria are the extracted quality factors of conceptual models and DB schemas during the study which are: completeness, complexity of the schema, data integrity, flexibility, comprehension, and implementation. Table 1 shows this analysis results.

According to Table 1, we notice that the global schema approach [20], the views approach [15], the frameworks approach [23], the variable schema approach [16], and VariaLBD guarantee database schema completeness. Moreover, the variable schema approach [16], the virtual decomposition approach [14], the approaches based on extended UML diagrams [19], [21], [22], [27], [28], and VariaLBD deal successfully with the database schema complexity. Table.1 shows that the majority of approaches do not effectively manage data integrity and variability in the same time. However, VariaLBD treats simultaneously the data integrity and the variability within the database schema since we considered the integrity constraints in the database and the dependencies and the cardinalities in the feature model during the VariaLBD definition to manage variability. In addition, VariaLBD and the tailored DBMS approach [24] favor database schema flexibility whereas the other approaches neglect working on this factor. We also find that VariaLBD is comprehensible by the user thanks to its standard formalism that gives it a universal aspect. Then, it is implementable and it has a tool support unlike the most of other approaches that do not have tools support or their tools suffer from many insufficiencies.

Table 1. Qualitative Analysis Results

| Quality factors / Approaches | Completeness | Complexity of the schema | Data integrity | Flexibility | Comprehension | Implementation |
|---|---|---|---|---|---|---|
| Global schema [20] | ++ | - | - | - | -- | - |
| Views [15] | ++ | -- | +/- | - | - | -- |
| Frameworks [23] | ++ | + | - | - | +/- | +/- |
| Variable schema [16] | ++ | ++ | ++ | +/- | + | +/- |
| delta-oriented programming approach [17], [18] | - | - | - | - | + | ++ |
| Tailored DBMS [24] | - | + | + | ++ | ++ | +/- |
| Virtual decomposition [14] | - | ++ | -- | + | ++ | + |
| Approaches based on extended UML diagrams [19], [21], [22], [27], [28] | +/- | ++ | +/- | ++ | ++ | -- |
| Database evolution in SPL [26] | - | +/- | ++ | - | ++ | ++ |
| Variable data model [25] | -- | - | -- | ++ | - | -- |
| **VariaLBD [5]** | **++** | **++** | **++** | **++** | **++** | **++** |

++ very good, + good, - bad, -- very bad

## 4.2. Quantitative Analysis of Experimental Results

To evaluate quantitatively our experimentation, we choose the view approach [15] and the variable schema approach [16] to implement their schemas for the quantitative evaluation as they are the closest approaches to VariaLBD. Then, we define measurable metrics which are CPU execution time, input/output cost, operator cost, and used size. We choose this metrics as they are the most pertinent when executing the DB schema. This evaluation is based on a comparison between metrics obtained from the execution of VariaLBD's DB schema and that obtained from the execution of the chosen approaches' DB schema. Therefore, in this evaluation we selected the following optional features: android, color, and radio for each

different schema and extract the metrics values related to this selection query. Table 2, Table 3, and Table 4 show the metrics values related to each selection query of each different feature for each different schema.

Table 2. The Metrics for the Photo Feature Selection Query

| Metrics / Approaches | CPU execution time | Input/output cost | Operator cost | Used size |
|---|---|---|---|---|
| VariaLBD | 0.000199 | 0.003125 | 0.003414 | 46B |
| View approach | 0.0003373 | 0.003125 | 0.0089671 | 53B |
| Variable schema approach | 0. 0003373 | 0.003125 | 0.0089671 | 53B |

According to Table 2, Table 3, and Table 4 the estimated values of the metrics CPU execution time, input/output cost, and operator cost related to the selection query of the features android, color, and radio. We notice that even changing the selected feature or DB schema that we worked with, the input/output cost remains the same for any approach. Moreover, the values of the CPU execution time and the estimated operator cost engendered by VariaLBD are always better than those engendered by the two other approaches regardless of the change of the selected feature or the DB schema generated by any approach. Hence, VariaLBD is the best performing thanks to the important time saving when the queries execution.

Table 3. The Metrics for the Product Feature Selection Query

| Metrics / Approaches | CPU execution time | Input/output cost | Operator cost | Used size |
|---|---|---|---|---|
| VariaLBD | 0.000199 | 0.003125 | 0.003414 | 70B |
| View approach | 0.0003345 | 0.003125 | 0.0074872 | 78B |
| Variable schema approach | 0.0003345 | 0.003125 | 0.0074872 | 78B |

Table 4. The Metrics for the Contact Feature Selection Query

| Metrics / Approaches | CPU execution time | Input/output cost | Operator cost | Used size |
|---|---|---|---|---|
| VariaLBD | 0.000152 | 0.003125 | 0.003476 | 26B |
| View approach | 0.0003461 | 0.003125 | 0.007841 | 37B |
| Variable schema approach | 0.0003461 | 0.003125 | 0.007841 | 37B |

According to Table 2, Table 3, and Table 4, we remark that the metric value is the same for the view approach and the variable schema approach. Otherwise, VariaLBD generate lower value even if we change the selected feature and the DB schema. Therefore, VariaLBD shows again her performance since it does not allow saving only time but also space.

## 4.3. Validity to Threat

We present in this paper a validity analysis [32] of the VariaLBD experimentation. The validity analysis enables proving that the experimental results have a satisfying validity for the chosen population of interest. So, we adapt the validity analysis proposed by [33]. To deduce conclusions, [33] proposed 4 types of threats to validity which are conclusion, internal, conceptual, and external.

### 4.3.1. Conclusion validity

VariaLBD evaluation is based on the following measures: the CPU execution time, the input/output cost, the operator cost, and the used size. We define these measures because they are objective. So, they are repeatable with the same result. Although these measures are more accurate because they are calculable but they are still influenced by several factors during the experimentation execution such as : the collected execution corpus size, the size of the studied SPL, the constraints number and its nature, and website traffic

to measure and analyze the performance. Thus, creating a dedicated tool support for the VariaLBD experimentation ensures a reliable processing implementation. The existence of this tool guarantees an independence between the treatments application and who realize the experimentation or on which it is realized. Therefore, the implementation of this experimentation is standard as the code does not inflect by the human judgment. Furthermore, his experimental material can be applied on different subjects and occasions.

#### 4.3.2. Internal validity

In our experimentation, data stored in our experimentation DB are not all collected from real SP product lines. Indeed, this does not badly affect the experimental results, but it fakes slightly the results of real data. That's why it's very important to study carefully the DB context to collect relevant data in order to ensure exact and conform modeling that in turn generates a variable schema tailored the users' needs.

In addition, in our experimentation there is no risk that the history affects the experimental results, only an intensional update of the source diagram initially entered by the user can modify the experimental results.

#### 4.3.3. Conceptual validity

In our experimentation, we apply VariaLBD on simple academic case studies. Therefore, VariaLBD must be tested on real-world scenarios in the future to handle with more complex SPLs and to ameliorate its generalizability.

#### 4.3.4. External validity

We define the following measures: CPU execution time, input/output cost, operator cost, and the used size for quantitative evaluation of VariaLBD experimentation results. Then, we use these measures to compare VariaLBD with other existing approaches. We note that if the experimental context changes then other measures can appear or disappear. For example, if the population interested by this experimentation or the experimental tools are modified, are these evaluation measures still sufficient for the VariaLBD validation, particularly in this experimentation because used tools are an individual choice?

## 5. Related Work

In literature, various approaches are proposed to model and implement variability in DBs. In this paper, we present the most relevant approaches which have the same objective as VariaLBD. Parsons [20] presented the global schema approach which has many drawbacks in terms of schema complexity, data integrity, flexibility, comprehension, and implementation. Moreover, it treats only the logical and physical schema at PSM level. Bolchini *et al.* [15] presented the view which produces views in addition to the global schema. This approach inherits the drawbacks of the global schema approach although the data integrity amelioration. Schäler *et al.* [16] presented an approach using the superimposition composition mechanism for generating automatically a variant of a particular information system which has an adapted schema. This approach improves data integrity, whereas the implementation of the DB schema become more complex. Besides, Herrmann *et al.* [26] proposed a tool support DAVE for adapting the schemas versioning technique to resolve the weaving problem in the DBs, but allowing only manual evolution and DB migration.

Note that the aforementioned approaches treat the variability only at the PSM level and they neglect the CIM and PIM levels. However some approaches are interested in these issues like the virtual decomposition technique approach proposed by Siegmund *et al.* [14] which proposes a virtual annotation of the conceptual DB schema in terms of features. Unlikely, this approach reduces the model's completeness and the data integrity and it treats the variability only at the PIM level. Then, Khedri and Khosravi [17], [18] managed the variability problem in DB schemas by using the delta-oriented programming, first at the logical and physical level and second at the conceptual level by proposing a meta-model at the CIM level. It's an evolutive and

implementable approach in contrast to the virtual decomposition approach [14], but it has various problems of completeness, flexibility, data integrity, and schema complexity. Moreover, the approaches based on extended UML diagrams [19], [21], [22], [27], [28] propose meta-models and variable conceptual models for DBs. Yet, its proposed models have insufficiencies such as they are not flexible enough to guarantee the DB evolution in SPL and they aren't implementable to test it in real-world scenarios. Finally, Abo Zaid and Troyer [25] proposed an approach for modeling data variability in data-intensive SPL. This approach enables the simplification of the data access process and the optimization of the storage space but without a tool support for supporting the mapping to the variable data model. So, it's not yet tested on real industrial scenarios.

## 6. Conclusions

In this paper, we present an overview of our contribution which is an approach to model and to implement variability in DB inherent to SPL (VariaLBD). Then, we experiment it on an industrial case study "smart phone product line". And we describe in detail this experimentation. The experimental results represent the VariaLBD's contribution which is the variable and relational DB schema. Next, we evaluate our experimentation. This evaluation consists of qualitative, quantitative, and validity to threat analysis.

Theoretically, we present in this paper two new contributions in software engineering. First, at the level of the DB evolution problem, VariaLBD proves that the generation of a variable DB schema ensures the DB evolution in a software system as the variable conceptual DB schema adapts to the features chosen by the users. Second, at the level of SPL engineering, VariaLBD adapts SPL for the modeling and the implementation of variability in the DBs. Our approach is inspired by the SPLs techniques for the variability management in the software systems' code to manage variability in DB schemas. Indeed, VariaLBD treats the variability in DBs inherent to SPL by considering a DBs set as SPL whose product is a particular DB. Hence, VariaLBD generates a variable DB schema adapted to the features desired by the client.

During experimentation, VariaLBD proves its performance. Unfortunately, this experimentation is carried out on a simple SPL that does not touch the depth of the industrial sector. Thus, more technical and complex scenarios can be considered to test VariaLBD in the future.

## References


[1] Weiss, D. M., & Lai, C. T. R. (1999). *Software Product Line Engineering: A Family-based Software Development Process.* Addison-Wesley Professional.

[2] Clements, P., & Northrop, L. (2001). *Software Product Lines.* Addison-Wesley.

[3] Batory, D., Sarvela, J., & Rauschmayer, A. (2004). Scaling step-wise refinement. *Proceedings of the IEEE: Vol. 30(6).Transactions on Software Engineering* (pp. 355-371).

[4] Czarnecki, K., & Eisenecker, U. (2000). *Generative programming: methods, tools, and applications.* Addison-Wesley.

[5] Khalfallah, N., Ouali, S., & Kraiem, N. (2016). Managing variability in database context using an MDE approach. *Proceedings of the 4th Int. Conference on Control Engineering & Information Technology (CEIT)* (pp. 1-6). Hammamet, Tunisia.

[6] Khalfallah, N., Ouali, S., & Kraiem, N. (2018). Approach for managing variability in database schema. *Journal of Asian Scientific Research, 8(6),* 221-236.

[7] Harris, P. (2002). *Designing, and reporting experiments in psychology* (2nd ed,). Open University Press: Buckingham.

[8] Flyvbjerg, B. (2006). Five misunderstandings about case study research. *Qualitative Inquiry, 12(2),* 219 – 245.

[9] Yin, R. K. (2002). *Case study research: design and methods.* Sage, Thousand Oaks, CA.

[10] Easterbook, S., Singer, J., Storey, M. A., & Damian, D. (2008). Selecting empirical methods for software engineering research. *Chapter 11 of the book Guide to Advanced Empirical Software Engineering* (pp. 285 – 311).

[11] Jedlitschka, A., Ciolkowski, M., & Pfahl, D. (2008). Reporting experiments in software engineering. *Chapter 8* Of The Book *Guide to Advanced Empirical Software Engineering* (pp. 201-228).

[12] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2000). *Experimentation in Software Engineering – An Introduction.* Kluwer Academic Publishers, Boston, MA.

[13] Khalfallah, N., Ouali, S., & Kraiem, N. (2018). Case tool support for variability managing in database schemas. *Journal of Software, 13(11),* 600-612.

[14] Siegmund, N., Kästner, C., Rosenmüller, M., Heidenreich, F., Apel, S., & Saake, G. (2009). Bridging the gap between variability in client application and database schema. *Proceedings of the GI-Fachtagung Datenbanksysteme für Business, Technologie und Web* (pp. 297 – 306).

[15] Bolchini, C., Quintarelli, E., & Rossato, R. (2007). Relational data tailoring through view composition. *Proceedings of the International Conference on Conceptual Modeling* (pp. 149 – 164). Springer.

[16] Schäler, M., Leich, T., Rosenmüller, M., & Saake, G. (2012). Building information system variants with tailored database schemas using features. *Proceedings of the International Conference CAiSE: Vol. 7328. Advanced Information Systems Engineering* (pp. 597 – 612). Springer.

[17] Khedri, N., & Khsoravi, R. (2013). Handling database schema variability in software product lines. *Proceedings of the 20th Asia-Pacific Software Engineering Conference (APSEC): Vol. 1.* (pp. 331 – 338).

[18] Khedri, N., &. Khosravi, R. (2015). Incremental variability management in conceptual data models of software product lines. *Proceedings of the Asia-Pacific Software Engineering Conference (APSEC)* (pp. 222 – 229).

[19] Ziadi, T. (2004). *Manipulation de lignes de produits en UML*. PhD thesis. Université de Rennes. France.

[20] Parsons, J. (2003). Effects of local versus global schema diagrams on verification and communication in conceptual data modeling. *Journal. Manage. Inf. Syst, 19(3),* 155-183.

[21] Clauß, M., & Jena, I. (2001). Modeling variability with UML. *Proceedings in GCSE Young Researchers Workshop.*

[22] Clauß, M. (2001). Generic modeling using UML extensions for variability. *Proceedings in Workshop on Domain Specific Visual Languages at OOPSLA.*

[23] Johnson. R., & Foote, B. (1988). Designing reusable classes. *Journal of Object-Oriented Programming, 1(2),* 22 – 35.

[24] Rosenmüller, M., Kästner, C., Siegmund, N., Sunkle, S., Apel, S., Leich, T., *et al.* (2009). SQL à la carte toward tailor-made data management. *Proceedings of the 13th conference GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW)* (pp. 117-136).

[25] Abo Zaid, L., & De Troyer, O. (2011). Towards modeling data variability in software product lines. *Proceedings of the 12th International Conference Enterprise, Business-Process and Information Systems Modeling (BPMDS) and Proceedings of the 16th International Conference, EMMSAD, CAiSE* (pp. 453-467). London, UK.

[26] Herrmann, K., Reimann, J., Voigt, H., Demuth, B., Fromm, S., Stelzmann, R., *et al.* (2015). Database evolution for software product lines. *Proceedings of the 4th International Conference on Data Management Technologies and Applications (DATA)* (pp. 125-133). Colmar, Alsace, France.

[27] Gomaa, H. (2005.). Designing software product lines with UML: from use cases to pattern-based software architectures. Addison-Wesley.

[28] Korherr, B., & List, B. A. (2007). UML 2 profile for variability models and their dependency to business processes. *Proceedings in DEXA Workshops* (pp. 829 – 834). ISBN: 0769529321, DOI: 10.1109/DEXA.2007.96.

[29] Moody, D. L., & Shanks, G. G. (1994). What makes a good data model? Evaluating the quality of entity relationship models. *Proceedings of the 13th International Conference: Vol. 881. Entity Relationship Approach* (pp. 94 – 111). Manchester, England.

[30] Moody, D. L.., & Shanks, G. (2003). Improving the quality of data models: empirical validation of a quality management framework. *Journal Information Systems, 28,* 619 – 650.

[31] Khalfallah, N., Ouali, S., & Kraiem, N. (2016). A proposal for a variability management framework. *Proceedings of the 7th International Conference on Sciences of Electronics, Technologies of Information and-Telecommunications (SETIT)* (pp. 133-138). Hammamet, Tunisia.

[32] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). Experimentation in software engineering. London: Springer Heidelberg New York Dordrecht.

[33] Cook T. D., & Campbell, T. D. (1979). *Quasi-experimentation: design and analysis issues for field settings.* Houghton Mifflin Company Boston.

**Nesrine Khalfallah** is a PhD student at RIADI Lab, ENSI, Campus of Manouba, Manouba, Tunisia. She obtained her maitrise degree in computer science applied to management in 2010 and his research master's in software engineering in 2012 and subscribed in the first year of thesis in 2014. She is working on this subject: the modeling and implementation of the variability in the database. She works in the public sector as a teacher and researcher.

**Sami Ouali** is an assistant professor in the College of Applied Sciences of Ibri in Oman. He is a member of the RIADI labs. His research interests lie in the areas of software engineering and software product line.

**Naoufel Kraïem** is a professor in the Department of Computer Science in Sultan Qaboos University. He is a member of the RIADI labs. His research interests include IT adoption and usage Information modeling, software engineering, software product lines, method engineering, web services and CASE tools.