Feature Models for Big Data Applications

Modeling Big Data Applications by applying Feature Models

Ioannis Zozas
Department of Informatics and Telecommunications
University of Western Macedonia
Kozani, Greece
izozas@uowm.gr

Stamatia Bibi
Department of Informatics and Telecommunications
University of Western Macedonia
Kozani, Greece
sbibi@uowm.gr

Dimitrios Katsaros
Department of Electrical & Computer Engineering
University of Thessaly
Volos, Greece
dkatsar@e-ce.uth.gr

Panagiotis Bozanis
Department of Electrical & Computer Engineering
University of Thessaly
Volos, Greece
pbozanis@e-ce.uth.gr

Ioannis Stamelos
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
stamelos@csd.auth.gr

***Abstract*—Modeling Big Data Applications is a key research topic for designing, analyzing, programming and deploying data-intensive applications, with high value and long-term trade-offs. The need for unified perspectives, architectures and requirements techniques is requisite. The current approach proposes the use of Feature Models to fill this gap by extending present model-driven engineering practices with utter purpose to define a reusable, extensible and highly configurable design approach for Big Data Applications.**

Keywords—big data; feature models; software engineering; model driven engineering;

I. INTRODUCTION

Big data has been a hot topic for both researchers and practitioners in the last decade. A chaotic-distributed explosion of data continues to flour every day our *data-driven world*, making more necessary than ever the need for advanced and efficient software and systems applications, that will collect, store, manage and analyze massive amounts of data.

Reports [5], [7] estimate that the Big Data Application (BDA) business will continue to grow over 50% in the following five-year forecast period. In addition, almost half of each enterprise's net will attract big data intelligence and analytics. On the other hand, an ineffective coordination of BDA strategies is pinpointed for ensuring successful application deployment. More than 55% of BDA implementation approaches do not adapt a clear road map, while over 74% fizzle to build their big data capabilities or operating models, take iterative approaches towards implementation, or even develop BDA competencies.

The above statistics prove that, in relation to the importance of big data, BDA analysis and design is far too important to be ignored. The unique nature of this type of applications differs from traditional applications due to the data-intensive nature they inherit. Until today, BDA-in-a-box is for now, far from a reality. The majority of practitioners and companies face a short-term syndrome by not having the discipline or even the patience to deal with a broader view of modeling [9]. The latter involves time consuming practices that lead to high costs on deployment. Moreover, current efforts proposed tend not to complement each other, while traditional analysis and design techniques fail in most cases to be applied efficiently [18].

The necessity to analyze and design BDA is based on the core key features of these type f applications: Data-intensive scalability, data distribution, functionality parallelism. In contrast to traditional application design, development and deployment, the present lack of one-time design to any-time deployment models, mature and consistent architectures, agile planning differentiations and deep focus on requirement analysis, strengthens BDA model engineering as a necessity to fully harvest the potential benefits of this reality. Like James Governor, founder of Redmonk cites on Twitter, "*data matures like wine but applications like fish*" [11].

In the current proposal, to fill the gap over BDA modeling, we propose a different approach based on the above mentioned

BDA key features and requirements, by applying Feature Models (FM). Feature Models consist of a hierarchical spanning tree of features [25]. Each feature is represented by a rectangle. Relations between features are shown as lines to form feature trees [13]. In this context, we focus on:

- A proposed generic approach for modeling BDA under the scope of Feature Models as a high-abstraction basis.
- A methodology for converting Feature Models into software design artifacts such as class diagrams.

Feature Models can support both changeability variability of BDA, which other proposed approaches do not cover fully due to the data variability factor [27]. The main advantages of this proposal is based on:

- Unification of diversity over current proposed models.
- Reduction of total cost over current deployments by requirements abstraction over agility differentiation.
- Avoidance of future inconsistencies that will prevent future expandability, based on the changeability that characterizes data-intensive applications.

The rest of the paper is organized as follows: In Section II we present the background of the problem (focusing on existing architectures and proposed techniques) and in Section III we describe in detail the proposed approach of applying Feature Models on BDA, In Section IV, we discuss related work so far in contrast to our proposal mentioning threats to validity, and conclude the paper in Section V.

II. Background of the problem

In this section we present basic attributes of BDA, describe architectures, requirement techniques and approaches proposed so far and present an overview of feature models.

A. The Big Data V's

In 2001, Doug Laney [16], was the first to define three basic attributes as the 3 V's of Big Data : *Volume*, *Velocity* and *Variety[16]*. Fifteen years later, new attributes were introduced leading to the multiplication of the V's defining big data. A short summary of this expansion is presented in Table I:

TABLE I. The Big Data V's

V type	V type definition and reference		
	Definition	***Introduced by***	***Date***
Volume	Size of the data set	Laney [16]	2001
Variety	Structured, semi-structured or unstructured data set types	Laney [16]	2001
Velocity	New data generated speed	Laney [16]	2001
Veracity	Data understandability quality	Hopkins [12]	2011
Variability	Statistical deviation of data	Hopkins [12]	2011
Value	Business value to be derived	Gartner [8]	2012
Viability	Implementation sustainability	Biehn [4]	2013
Volatility	Available data change pace	Kahn [14]	2014
Vitality	Importance of certain data sets in the data pool	Accenture [1]	2014
Visual	Representation of data	Rijmenam [19]	2015
Visibility	Actionable data type	Rijmenam [19]	2015
Vinculatiry	Connectivity of data sources	Rijmenam [19]	2015

While the above list continues to expand, the original target when dealing with Big Data still remains the same: To extract value and knowledge from diverse data sets, without shaking the original foundation of the early four V's [10].

B. Architectures and requirement techniques for BDA

In 2008, Gorton [10] addressed the challenges when developing data intensive applications and supporting infrastructures. Based on real world examples, such as astronomy and social computing, proposed the application of dynamic design principles and adaptive architectures as a solution to configurable architectural properties.

In 2012, Begoli [3] reviewed the appliance of currently available architectures over BDA modeling and also provided a summary of all relevant requirement techniques. In addition, he outlined the most common platforms for knowledge discovery from data, and recorded their architectural properties. Begoli [3] finally derived empirical architectural principles based on his experience over governmental data analysis.

In 2015, Anderson [2] proposed an agile life-cycle architecture to match existing frameworks with iteratively defined requirements, and provided a case study on BDA for social media. He identifies and discusses challenges and the corresponding solutions associated with the design and development of data-intensive software systems On the basis of Twitter data analysis, he proposes persistent software architecture. Gomez [9] applied UML for modeling BDA and proposed the use of stochastic petri nets for requirements specification. Based on three different application domain case studies (fraud detection, social sensor news acquisition, vessel traffic management), the proposed models were verified.

Also in 2016, Xing [23] proposed the use of iterative modeling techniques on architecture design and requirements specifications. Also provided a case study over BDA machine learning operations. He focuses on exploring new design strategies by revisiting traditional architecture principles.

While most appliances of the above proposed methods proved to be quite effective, they fail to describe mostly the appliance variability of BDA [19] :

TABLE II. Proposed BDA design and analysis approaches

Architectures	***Requirements focus***	***Appliance***	***Reference***
Adaptive behavior	Predictable requirements	Astronomy, Social computing	2008 [10]
Match-making	A priori requirement specification	US state, federal data analysis	2012 [3]
Agile life-cycles	Matching frameworks with requirements	Social Media	2015 [2]

Architectures	*Requirements focus*	*Appliance*	*Reference*
UML models	Stochastic Petri nets	Fraud detection, Social sensor news, Vessel traffic	2016 [9]
Iterative modeling	Iterative requirements specification review	Social Media	2016 [23]

C. Feature Models

Feature Models as a modeling technique, is considered to be one of the very first concepts of Software Engineering and Domain Analysis areas, introduced in 1990 [13]. By consisting of simple hierarchical models, they enable capturing the commonality, variability and features of software product lines, and thus, are considered to be a prevalent approach to model products. Software Product Lines (SPL) are defined as a set of software-intensive systems that share a common managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [25].

Feature Models are considered as a strategic concept for Software Product Lines in order to reduce the overall cost of software manufacture, by enabling software reusability and defining optimized software components [13][30]. This concept has been widely applied in software industry especially in application domains that present highly diversified requirements. Quinton [28] proposes feature modeling for mobile application development, Sinz [29] for artificial intelligence configurations in automobile industry, while Danfoss Drives produces industrial frequency converters by using feature modeling [30].

The feature tree is the core of Feature Models, depicting in a visually manner all features of an application by using groups of increasingly levels of detail and functionality, in a form of a hierarchical diagram. A tree-like structure links all features by using variability relationships, and optionally, cross-tree constraints connect features [21]. Generic feature models propose four variability and two constraint relationships [22] :

- *OPTIONAL* - A feature can be included or excluded (noted as a white circle on top of a feature)
- *MANDATORY* - A feature must be included (noted as a black circle on top of a feature)
- *OR* - One or more features from a group of child-features can be included
- *ALTERNATIVE* - Exactly one feature from a group of child-features must be included
- *REQUIRE* constraint - A feature is included while another feature must be also included.
- *EXCLUDE* constraint - A feature is included while another feature must be excluded.

A simple example is depicted in Figure 1 for data Storing and Processing. Two mandatory feature groups (Storing and Processing collecting) as well as three mandatory features and one OR sub-features are included. The BDA critical Analytics processes [15] (e.g. Processing and Analytics models [13]), based on various Analytics techniques (e.g. Machine learning [13]), require data to be processed, mainly stored in non relational NoSQL databases [3] with optional relational databases use [14].

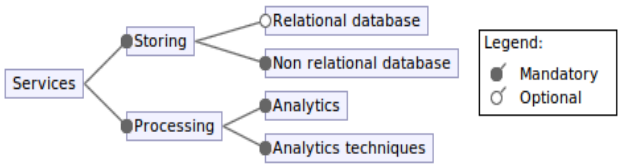

Fig. 1. A BDA services model

TABLE III. BDA ANALYTICS CONFIGURATION MATRIX

Relational database	*Non relational database*	*Analytics*	*Analytics techniques*
X	X	X	X
	X	X	X

Table III depicts the configuration matrix for the above Feature Model. Two configurations derive on the basis of the defined constraints. Each feature of the proposed Feature Model tree can further expand to include new feature and constraints. This leads to new expanded configuration matrices, while the complexity of the solution space increases.

The significance variability of Big Data V's characteristics for each BDA application, differs over the expected results of each deployment [14]. Each deployment presents lead eventually to a collection of similar applications that include similar sets of software assets. The result is the formation of a virtual software product line satisfying the need for BDA deployment.

To properly model the Table I V's, Feature Models can describe this variability that is not of the compositional type but rather of potential properties. Out of the above Table II method, Feature Models excel due to coarse grain variability nature of BDA features [21], and a comparison to other proposed techniques on low V's characteristics satisfaction is depicted on the following table :

TABLE IV. BIG DATA V'S MODELING TECHNIQUES SATISFACTION

Technique / V types	Volume	Variety	Velocity	Veracity	Variability	Value	Viability	Volatility	Vitality	Visual	Visibility	Vinculatiry
Adaptive behavior [1][10][19]	X	X	X	X	X	X	X	X	X	X		
Match-making [1][3][4][14]	X	X	X	X	X	X				X	X	X
Agile life-cycles [1][2][12]	X	X	X		X	X	X	X		X	X	X
UML models [1][3][4][9][14]	X	X	X	X	X	X				X	X	X
Iterative modeling [4][16][23]	X	X		X	X	X		X	X	X	X	X
Feature models [1][15]	X	X	X	X	X	X	X	X		X	X	X

III. Building the BDA Feature Model

A. Defining the features

Features are abstract concepts for describing commonalities and variabilities [13],[17]. These represent characteristics of a system relevant for some Stakeholder, where depending on the interest of the latter, can be requirements, technical functions, function groups or quality characteristics. BDA applications serve data-intensive scalability, data distribution, functionality parallelism. On the basis of Kune's older anatomy of a BDA [15] and Pääkkönen's architectural reference over modern big data services [26], all necessary features are divided into physical (Infrastructure) and logical (Services) groups.

The first key concept represent the ability to physically collect (e.g. via sensors), transfer (e.g. via network) and store data (e.g. in hard disk clusters). The second key concept represents the ability to logically collect (e.g. via crawlers), process (e.g. via data normalization) and analyze data (e.g. via analytics or machine learning algorithms). These represent basic features BDA include, with possible further expansion.

The feature definition based on the above feature groups is summarized on the following table, based on our literature research and mostly over Kune's [15] wide accepted proposal :

TABLE V. Feature definition

Feature Group / Feature	***Description***
Infrastructure / Network	Hardware interconnection of the system with a) internal components, b) external world [15][14][17]
Infrastructure / Storage	Hardware physical or virtual storage of data, used to store either temporary or permanently data [15][10]
Infrastructure / Data sources	Data sources that feed the system with data, mostly external of the system [15]
Services / Collecting	Collecting process from carious data sources and transferring them to feed the system [15][23]
Services / Storing	Storing process of the collected data and distribution between available storing spaces [15][14]
Services / Preprocessing	Prepare data for analysis by transferring from and to storage, compressing and cleaning [15][23]
Services / Processing	Data analysis to deliver knowledge from big data [15] [23]
Services / Interface	Visualization of the data analysis results, and interaction between end users with the system [15][18]

B. Building the Feature Model

The final proposed General Feature Model is shown in Figure 2. The features depicted represent potential system variants of a BDA. Since this generic model represents coarse grained functionality, no constraint relationships are described as all abstract features can be further expanded.

On the infrastructure part feature groups :

- The *data network* group must rely over *cable* network, and optionally *wireless* network (to support distant environmental sensors data collections).
- The *data storage* group must be relied over *network storages* (expanded to cloud storage or network clusters) and optionally to *direct attached storages* for temporary data collection purposes.
- The *data sources* group alternative supports *hardware* (sensors, embedded devices, actuators) or *software* (application logs, web crawlers, scrappers, monitor agents or bulk collectors), according to the defined data sources of the final deployment.

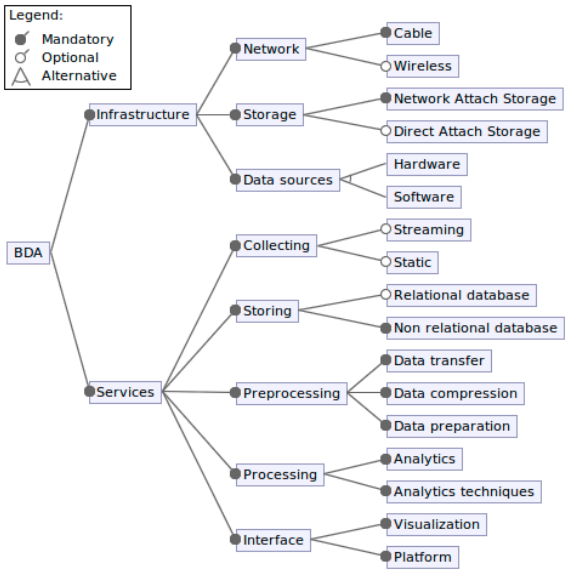

Fig. 2. *A BDA general Feature Model*

On the Services part, feature groups are modeled as part of a BDA value-chain operational diagram :

- The *collecting* group includes techniques and methods that operate on the infrastructure to collect data, whether by *streaming* (either real time or batch operations) or *statics* collection processes.
- The services *storing* group includes logical features that rely over the hardware – database structures and storing techniques. *Non relational databases* are always used to store huge amounts of data, while optionally *relational databases* used for result storing.
- The *preprocessing* group includes data extraction, transfer, distribution, compression, preparation, validation, and transformation operations. The group contains the most time-consuming and critical features for the validity of the input data to be analyzed. *Data transfer* deliver data from sources to the system, *Data compression compasses* the data before deliverance and *Data preparation* alters data structure via cleaning techniques before processed.
- The *processing* group is the core of data analytics. Knowledge derives from raw data through processing, *analytics* models, analytics services and *techniques*.
- The *interface* group includes *visualization* (how derived knowledge through data analytics is represented to the end user, and how an interaction link will be established) and *platform* features (how the end-product will be operated by the end-users.

The above feature groups are abstract in the tree structure, and expand further to non-abstract features in larger depths of complexity [15], [20]. Cardinality relationship can also be applied over the expanded model. As a result based on the above conversion cases, *a UML integration to transmute Feature Models to UML models is feasible*. By this transmutation, *automatic generation of readable and maintainable code can be achieved*. The above proposal combines Feature Models variability, Use Case Diagrams interactions, and Class Diagram conceptual extensibility.

C. Deriving the Use Case Diagram

Use Case Diagrams are defined as behavior diagrams that are used to describe a set of use cases, that a system can perform with external actors of the system [22]. The described constraint reproductivity is possible to be converted to Use Case Diagrams, by representing use sequence of actions, to provide a measurable value, and system usage requirements.

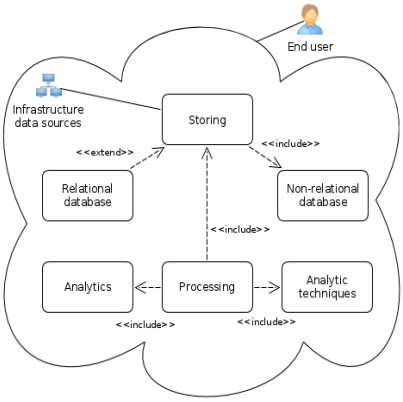

Fig. 3. A BDA processing and storing use case diagram

For demonstration purposes, consider the Feature Model *Analytics* sub tree based on Figure 1. This model extends the Storing and Processing features as described on Figure 2. A generic example of this conversion of analytics example is depicted in Figure 3.

The use case lists totally six *event steps*, two *primary stakeholders* (End user and Infrastructure data sources). The *postcondition success guarantee* is to deliver data analysis results to the End user (either by visualisation or other data presentation steps, irrelevant to the current scope). Similar, the *Precondition* to storing step inputs data from Infrastructure data sources stakeholder. The use case *Trigger* invokes data storing process.

As for the *Basic Flow,* the Storing step includes storing and access to Non-relational databases as a primary data storage, but also extends to Relational databases as an option, in correspondence to figure 1 feature tree. BDA concern huge amounts of data nearly all stored in huge non-relational, and a small percentage or none, to relational databases [10] The Processing stem includes Analytics and Analytic techniques steps to perform data processing. The most noticeable interaction is the inclusion of Storing by the Processing step. In order to process data, access to storage is mandatory.

Taking advantage of a BDA near linear use case operation (collecting, storing, processing, analysing and visualizing data [15]), fully derivation to a use case diagram can be achieved by deriving inclusion steps from mandatory features, and extension steps from optional features. Concerning the figure 1 feature tree, for the case of the Services feature group, the expected derivation can be achieved based on this linearity, as use case *Extensions*. Concerning the Infrastructure feature group, inclusion or extension relations to Services feature groups can be defined as needed.

Use Case Diagram can be further expanded to include optional or mandatory use case nodes, by converting features to use cases. Also, the circumvention of the whole diagram is excluded as it requires more expansion deriving from the expanded feature tree.

D. Deriving the class Diagram

As a result from Feature Model technique and the Use Case Diagram, a UML Class Diagram conversion is feasible. Class Diagrams are the main blocks of object-oriented modelling, to represent both main elements, interactions and classes to be programmed [23]. Most common practices involve [27] :

- Encoding of Feature Models to class models by converting child-less features (features that cannot be extended) to attributes and parental features to classes.
- Encoding sub-feature relationships either as property nesting or as UML ***composition***.
- Encoding ALTERNATIVE relationship to UML ***generalizability***.
- Correspond feature multiplicities to property multiplicities, and XOR constraints to enumerations.

A generic example of this conversion of analytics example is depicted in Figure 4 :

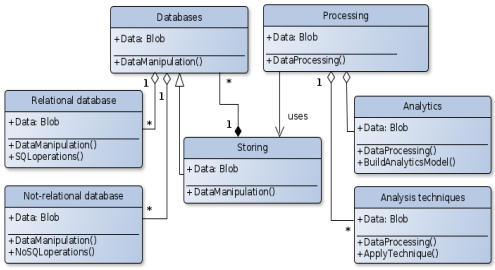

Fig. 4. *A BDA Storing and Process class diagram*

Data is represented as Blobs due to the variability nature of input data structured to a BDA [27]. Based on the above use case diagram, six classes are introduced. Databases as parental class aggregates one-or-more relational and non-relational databases classes as foretold to satisfy data storage requirements [10]. Both these types of databases serve data storing and retrieving, operational functionality differs on a SQL and NoSQL basis. A set of databases classes composited an introduced Storing master class to unify operations (one-to-many) over different sets of databases.

Processing as parental class aggregates Analytics subclass for build the data analytics model. Also aggregates one-or-more Analytics Techniques subclass for deploying one or more techniques, based on the family of techniques required for data analysis [15]. Last but not least, Processing class associates with Storing class to derive the mentioned inclusion use case action.

By converting each feature to Use Case Diagrams and extending each to Class Diagrams, the full conversion of a Feature Model is feasible. An empirical application should define the efficiency, complexity and computation effort of this conversion. All Class Diagram features can can be derived through this process to be fully described.

IV. Conclusions

In this paper, we have proposed a Feature Model-based approach for modeling BDA. We have demonstrated how Feature Models could be used in order to guide the system variability depending on the unique nature of this type of applications, and a way of converting Feature Models to UML models. Modeling BDA is appraised as a high value but also as a long trade-off topic. *Our approach fills the gap in modeling BDA by proposing feature-oriented design to superimpose requirements to BDA components, by taking special consideration over variability and expandability to this specific family of applications. Also highlight the value of applying Feature Modeling compared to other modeling techniques as more effective and suitable over BDA.* Appliance, customization and deployment variability, dwell deep in the heart of big data nature, which is data variability. The suggested approach focuses on overall modeling BDA as a different approach from previous research efforts that mostly focus on single case appliances [19][27].

The usefulness and practicability of the suggested models can only be proven by conducting empirical case studies and statistically significant tests using real-life data from existing libraries for reusable components. The empirical evaluation of the proposed model is planned to be presented as future work.

References

[1] Accenture, “Companies Are Satisfied with Business Outcomes from Big Data and Recognize Big Data as Very Important to Their Digital Transformation”, Accenture Study Shows Newsroom, https://accntu.re/2rnebEW. [Accessed: 22-Apr-2017]

[2] Anderson, K. M., “Embrace the Challenges: Software Engineering in a Big Data World”, in: 2015 IEEE/ACM 1st International Workshop on Big Data Software Engineering (BIGDSE)', pp. 19–25, 2015.

[3] Begoli, E. (2012), A Short Survey on the State of the Art in Architectures and Platforms for Large Scale Data Analysis and Knowledge Discovery from Data, in 'Proceedings of the WICSA/ECSA 2012 Companion Volume', ACM, New York, NY, USA, pp. 177--183.

[4] Biehn, N., “The Missing Vs in Big Data: Viability and Value - Innovation Insights”, http://bit.ly/2qAwtl9 [Accessed: 22-Apr-2017]

[5] Capgemini, “Cracking the data conundrum: How successful companies make big data operational,” Capgemini Consulting Worldwide, http://bit.ly/1E27l94 , [Accessed: 22-Apr-2017]

[6] DeLine, R. (2015), Research Opportunities for the Big Data Era of Software Engineering, in 'Proceedings of the First International Workshop on BIG Data Software Engineering', IEEE Press, Piscataway, NJ, USA, pp. 26—29.

[7] Forbes, “Forbes Insights: The Big Potential of Big Data.”, http://bit.ly/2qCZKvM. [Accessed: 22-Apr-2017]

[8] Gartner, http://gtnr.it/1RuD6gU. [Accessed: 22-Apr-2017]

[9] Gómez, A., et al. (2016), Towards a UML Profile for Data Intensive Applications, in 'Proceedings of the 2Nd International Workshop on Quality-Aware DevOps', ACM, New York, NY, USA, pp. 18—23.

[10] Gorton, I. (2008), Software Architecture Challenges for Data Intensive Computing, in 'Seventh Working IEEE/IFIP Conference on Software Architecture, 2008. WICSA 2008', pp. 4—6.

[11] Governor, J., “Why Applications Are Like Fish and Data is Like Wine,” James Governor’s Monkchips. http://bit.ly/1THbjJS. [22-Apr-2017]

[12] Hopkins, B. and Boris, E. (2011). Expand your digital horizon with big data, Forrester Research Inc, http://bit.ly/2r1D89i. [22-Apr-2017]

[13] Kang, K., et al., (1990), Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-021, Software Engineering Institute, Carnegie Mellon University, Pittsburgh.

[14] Khan, M.A., Fahim Uddin M.,Gupta N. (2014), “Seven V’s of Big Data”, Proceedings of 2014 Zone 1 Conference of the American Society for Engineering Education (ASEE Zone 1), Bridgeport, Connecticut, USA, pp.3-5.

[15] Kune, R.; Konugurthi, P. K.; Agarwal, A.; Chillarige, R. R. & Buyya, R. (2016), 'The anatomy of big data computing', Software: Practice and Experience 46(1), 79–105.

[16] Laney, D. "3D Data Management: Controlling Data Volume, Velocity and Variety", Gartner, http://gtnr.it/1bKflKH. [Accessed: 22-Apr-2017]

[17] Ludewig, J., (2003) “Models in software engineering – an introduction,” SoSyM, vol. 2, no. 1, pp. 5–14.

[18] Madhavji, N. H.; Miranskyy, A. & Kontogiannis, K. (2015), Big Picture of Big Data Software Engineering: With Example Research Challenges, in '2015 IEEE/ACM 1st International Workshop on Big Data Software Engineering (BIGDSE)', pp. 11—14.

[19] Rijmenam, van M., “Why The 3V’s Are Not Sufficient To Describe Big Data”, Datafloq, http://bit.ly/1FnnliR. [Accessed: 22-Apr-2017]

[20] Runeson, P., Höst, M., “Guidelines for conducting and reporting case study research in software engineering,” Empir Software Eng, vol. 14, no. 2, p. 131, Apr. 2009.

[21] Usman M., Iqbal M. Z., Khan M. U., (2017), ‘A product-line model-driven engineering approach for generating feature-based mobile applications’, Journal of Systems and Software, issue 123, 1–32

[22] Vachharajani, V., Pareek, J., “A Proposed Architecture for Automated Assessment of Use Case Diagrams,” International Journal of Computer Applications, vol. 108, no. 4, pp. 35–40, Dec. 2014.

[23] Vargas, R. T., Nugroho, A., Chaudron, M., Visser, J. “The Use of UML Class Diagrams and Its Effect on Code Change-proneness,” in Proceedings of the Second Edition of the International Workshop on Experiences and Empirical Studies in Software Modelling, New York, NY, USA, 2012, p. 2:1–2:6.

[24] Xing, E. P., et. al., “Strategies and Principles of Distributed Machine Learning on Big Data”, Engineering 2(2), 179–195, 2016

[25] P. A. da Mota Silveira Neto, et. al., “A systematic mapping study of software product lines testing,” Information and Software Technology, vol. 53, no. 5, pp. 407–423, 2011.

[26] P. Pääkkönen and D. Pakkala, “Reference Architecture and Classification of Technologies, Products and Services for Big Data Systems,” Big Data Research, vol. 2, no. 4, pp. 166–186, 2015.

[27] K. Bąk, et al., “Clafer: unifying class and feature modeling,” Softw Syst Model, vol. 15, no. 3, pp. 811–845, Jul. 2016.

[28] C. Quinton, S. Mosser, C. Parra, L. Duchien, “Using Multiple Feature Models to Design Applications for Mobile Phones,” MAPLE / SCALE workshop, colocated with SPLC’11, Munich, Germany, 2011, pp. 1–8.

[29] Sinz C., Kaiser A., Kuchlin W., “Formal methods for the validation of automotive product configuration data”, Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 2012, 17 (2): 75 – 97.

[30] H. P. Jepsen, D. Beuche, “Running a Software Product Line: Standing Still is Going Backwards,” in Proceedings of the 13th International Software Product Line Conference, Pittsburgh, PA, USA, 2009, pp. 101–110.