A Tool-Supported Approach for Modelling and Verifying MapReduce Workflow Using Event B and BPMN2.0

Mayssa Bessifi[1], Ahlem Ben Younes[2] and Leila Ben Ayed[3]
[1]*FSEGS, University of Sfax,Tunisia*
[2]*ENSIT, University of Tunis, Tunisia*
[3]*ENSI, University of la Manouba, Tunisia*

Keywords: Big Data, MapReduce, BPMN2, Event B, Formal Verification, Data Quality Properties, Refinement.

Abstract: Big data techniques are increasingly applied in critical applications (such as health, marketing nuclear research field, aeronautics field), so it is desirable that a systematic method is developed to ensure the correctness of these applications. As an aid to designers and developers, we propose a model-driven approach for the specification and formal verification of MapReduce workflow applications using a semi-formal language which is BPMN2 to represent MapReduce workflow and the Event B method for analysis. Our approach starts with the graphical modelling of the MapReduce application as a chain of MapReduce design patterns using an adapted BPMN2 notation. Then the model is transformed into an equivalent Event B project, composed by a set of contexts and machines linked by refinement, that can be enriched with a variety of desirable properties. The approach has been automated using a set of mapping rules implemented in a first prototype tool. We illustrate our approach with a case study "Fireware" and we verify data quality properties such as data non-conflict and data completeness.

1 INTRODUCTION

Nowadays, zettabytes of data are generated every day. This flood of data is accessible in all formats, i.e. structured, unstructured and semi-structured. The large volume, the variety and the rising speed of generation of this huge amount of data give rise to the term Big Data (S and Ravishankar 2019). In order to store, process and analyze such data new technologies and storage mechanism are required. One of the most successful paradigms is MapReduce. It has been proposed by google and emerged as the core of several Big Data frameworks such as Hadoop, spark and Storm, etc.

Hadoop is developed by Apache Software Foundation as an open-source framework. It is the leader implementation of the MapReduce paradigm to solve big data problems. Hadoop cluster is mainly composed by two layers: a Hadoop distributed file system called HDFS (Rehan and Gangodkar 2015) for storing data and Hadoop MapReduce paradigm for parallel processing. The Hadoop system is highly scalable; i.e., it can combine the computing and the storage power of thousands of computers to stock and process the massive data in a distributed way.

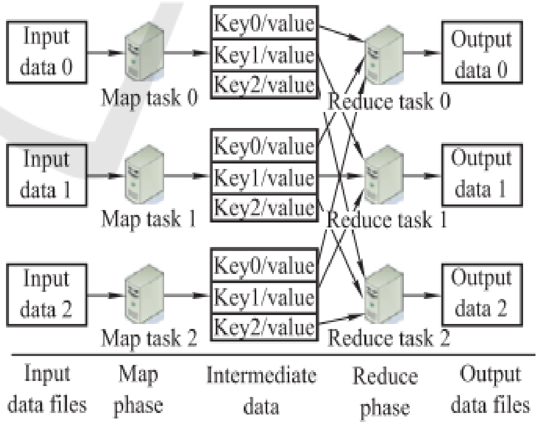


Figure 1: Procedure of a MapReduce job.

As depicted in Figure 1, a MapReduce job is created in two phases: map and reduce. The map phase, composed of parallel map tasks, receives raw data as input and generates a set of key-value pairs. Any given "key" can be used by several pairs. Then, the reducer, which can be one or many parallel reduce

Bessifi, M., Ben Younes, A. and Ben Ayed, L.
A Tool-Supported Approach for Modelling and Verifying MapReduce Workflow Using Event B and BPMN2.0.
DOI: 10.5220/0012861300003753
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 19th International Conference on Software Technologies (ICSOFT 2024)*, pages 504-510
ISBN: 978-989-758-706-1; ISSN: 2184-2833
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

tasks, processes pairs from mappers and aggregates values with the same key. The output of the reducer can be written in HDFS or send to another Mapper, in the case of MapReduce workflow.

A MapReduce workflow processes terabytes of data through MapReduce jobs connected through producer-consumer relationships. Each job consists of one Map phase and one Reduce phase. A phase includes multiple parallel Map or Reduce tasks.

Although the Hadoop MapReduce framework is easy to grasp, the development of complex MapReduce workflow can be a tedious task and require the collaboration of many developers. The intervention of different developers raises the possibility of mistakes and bugs, in the map and reduce programs, that can interrupt the MapReduce execution or produce inaccurate output. Also, as MapReduce applications are like every other application, from the design phase a set of requirements are envisaged and need to be verified from an early stage to reduce the possibilities of failure or inconsistency. Therefore, we propose in this paper an approach driven by MapReduce design patterns, for the modeling and formal verification of MapReduce workflow using both the standard BPMN2(Correia and Abreu 2012) and the Event B method(Abrial 2010).

The objective of the proposed approach is twofold. First, the approach is developed to help designers to easily design their MapReduce workflow using the graphical tool of a well-known and rich standard which is BPMN, and based on a set of predefined MapReduce design patterns.

Second, by automatically transformed the MapReduce workflow to a formal notation, Event B, the designer can make further analysis and consequently detect any errors at earlier stage.

The rest of the paper is organized as follows: section 2 presents related works and our main contributions. Section 3 presents briefly BPMN and the Event B method. Section 4 describe the proposed approach for the specification and verification of MapReduce workflow. Then a case study is presented. Finally, section 6 concludes the paper.

2 RELATED WORKS

Although the wide spread of big data applications, Only 13% of organizations have achieved full-scale production for their Big Data implementations as mentioned in the research (Colas et al. 2014) at Capgemini Consulting. The main reason behind this problem is the technical difficulty to design and develop effective big data applications. Hence, in order to increase the productivity in the development of Big Data applications, new languages, methodologies and tools needed to be created for assisting and guiding developers. In this context, a number of research are realized(Perez-Palacin et al. 2019)(Bersani et al. 2019)(Lim, Herodotou, and Babu 2012).

In (Chiang et al. 2021), Chiang et al. adopts petri net to visually model the MapReduce framework in order to verify its reachability property. The study act as guidelines for the developer to ovoid common errors such as when the system can't find input or output file.

Another variation of petri nets, Prioritized–Timed Coloured Petri Nets, is used in (Ruiz, Calleja, and Cazorla 2015), to evaluate the performance of the application "SentiStrength" for the Hadoop module in cloud environments. Simulations are realized by CPNtools to find out the best performance–cost agreements in advance.

Zeliu et al. analyses the rationality of MapReduce workflow by using the Object Petri Net(Hong and Bae 2000) in (Zeliu et al. 2019). The rationality criteria are defined by: the absence of a strangler task, the absence of conflict map and the reasonable execution time.

In the above cited works, petri net with its different variation is adopted at the design phase of MapReduce application. Although, it is a mature formalism that is widely used , a petri net model can become very complex(Bessifi, Younes, and Ayed 2022). Also, the use of the model checking technique as a verification technique pose always the problem of the growing number of explored states and thus the time of verification of data intensive applications such as MapReduce applications.

In (Zhang et al. 2020), a runtime verification approach at code level is presented. Both Map and reduce programs are written in MSVL(Wang et al. 2020) language, properties to verify are expressed in PPTL(Duan et al. 2019) formulas then verified using the MSVL model checker. Two case studies are presented to verify several data properties.

In this paper, we propose a model-driven framework for the specification and verification of MapReduce workflow to help developer to create a correct by construction MapReduce application. This approach combines the power of two different languages: Event B and BPMN, to provide a prototype tool that can be used to easily create high quality applications.

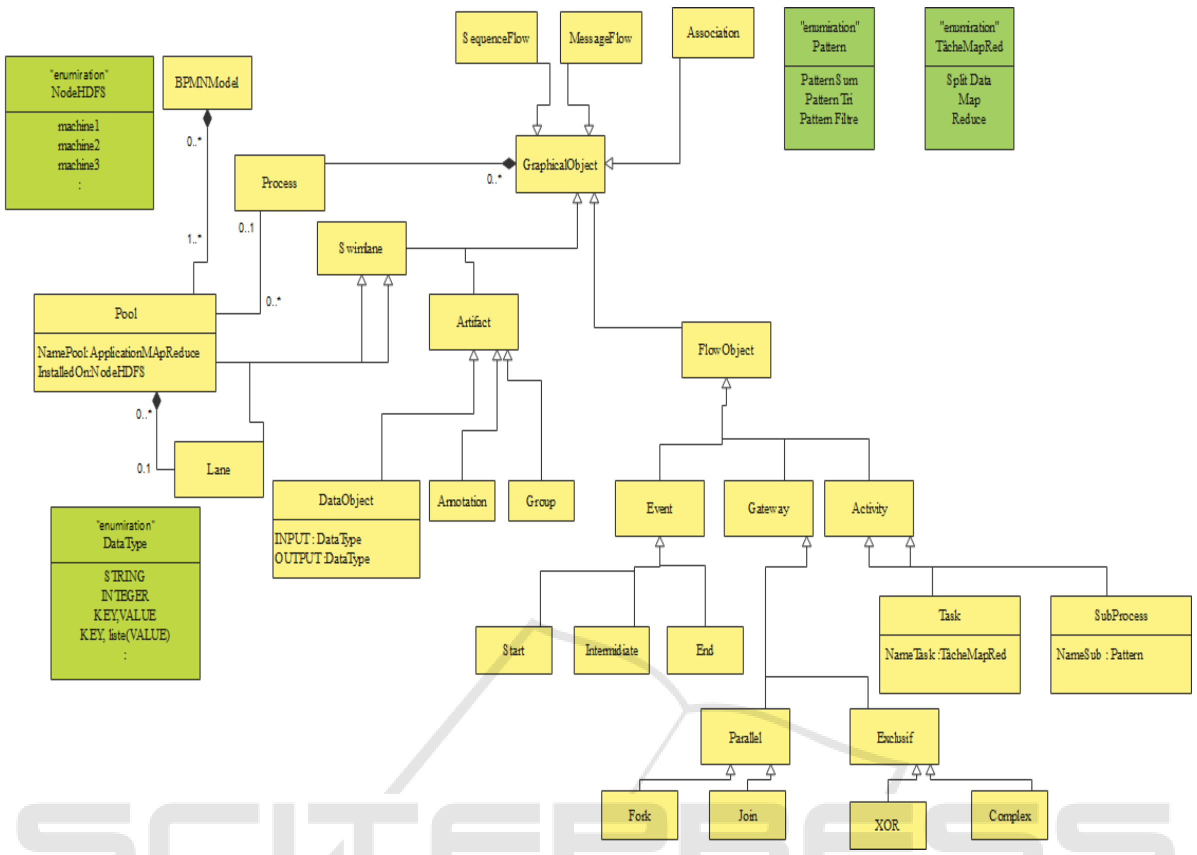


Figure 2: BPMN2.0 meta-model adapted to MapReduce workflow.

The originality of our contribution consists of:

- Reducing designers' efforts due to the use of BPMN graphical notation to define the proper MapReduce workflow as a chain of MapReduce design patterns.
- The automatic meta-model transformation of the MapReduce workflow to an equivalent formal description, through a set of mapping rules implemented using the Kermeta model transformation language.
- We adopt Event B as our target formal language, supported by its RODIN tool, because of its expressivity to model a complex system at different levels of abstraction thanks to the refinement concept, and the verification technique is based on theorem proving which does not suffer from the explosion number of states.

3 BACKGROUND

3.1 BPMN2.0

The BPMN “Business Process Model and Notation” is a standard business process model and notation developed by the Object Management Group. The main goal of BPMN is to provide an easily to understand notation that is capable to represent a semantically complex process.

BPMN2 is a semantically rich modelling language. While a UML activity diagram has about 20 different modelling constructs, a BPMN2 process model diagram (the most complex of the 3 available) has about 100 different modelling constructs, including 51 event types, 8 types of gateway, 7 types of data, 4 types of activities, 6 activity markers, 7 types of tasks, 4 types of flows, basins, corridors, etc(Correia and Abreu 2012).

3.2 Event B Method

Event B is a formal method based on mathematical foundations (first-order logic and set theory). An Event B model is built by two types of components: contexts and machines that respectively represent the static and dynamic behavior of the model. They are identified by a unique name and consist of a set of clauses.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

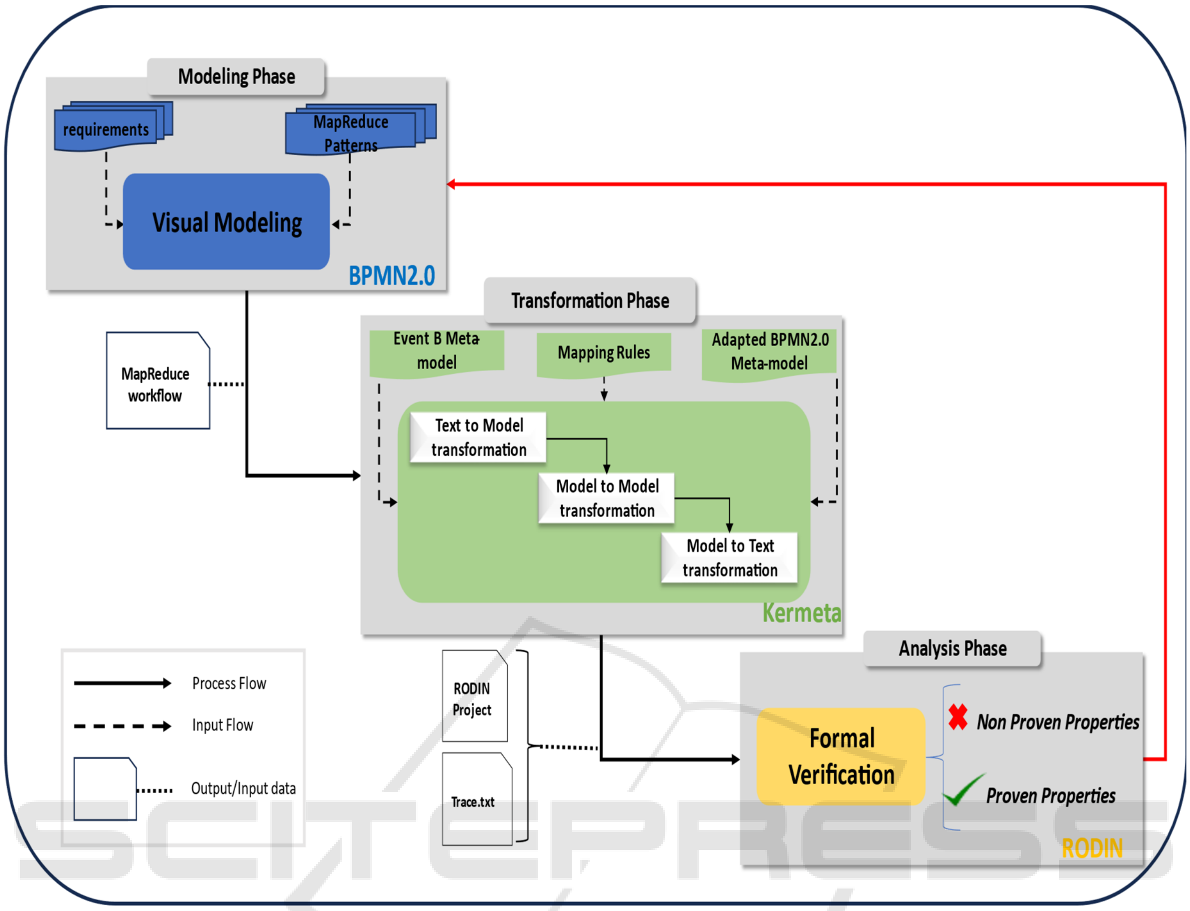


Figure 3: Overview of the proposed approach.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

A powerful feature of Event B is the notion of refinement. The user starts the modelling process with a very simple abstract model that can be gradually refined into something that is close to the implementation of the system.

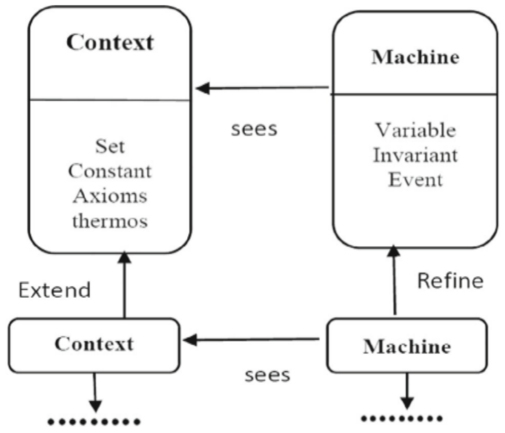


Figure 4: RODIN Project architecture (Toman et al. 2024).

4 PROPOSED APPROACH

As illustrated in Figure 3, the proposed model-based approach is based mainly on three major phases. It begins by designing the MapReduce workflow as a sequence of MapReduce jobs (each job is a MapReduce design pattern) using the BPMN tool, Camunda. After the 1 to the proposed BPMN meta-model. Second a model-to-model transformation is executed to generate an Event B model conforms to the Event B meta-model. Finally, the generated Event B model is transformed into a RODIN project by the means of a “Model to Text” transformation.

Once the RODIN project is opened, the designer can add properties, expressed as predicates in the different clauses (invariant, guard and theorem). All typing properties as well as data-related properties, such as data completeness and data non-conflict, are automatically generated by the transformation process.

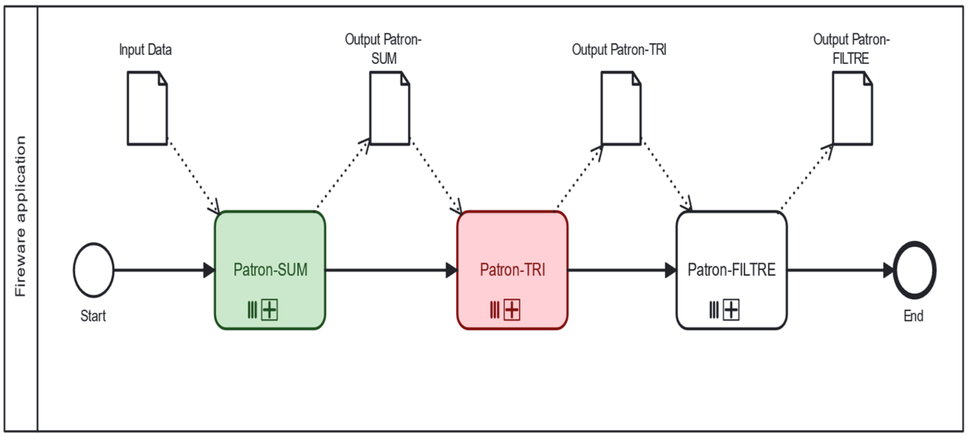
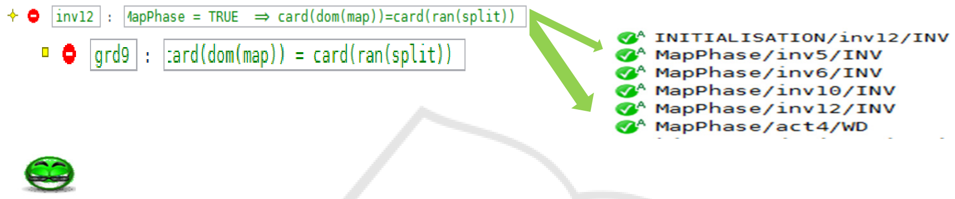


Figure 5: Graphical modelling of a MapReduce "Fireware" application with BPMN2.0.



```
inv12 : ⁄apPhase = TRUE  ⇒ card(dom(map))=card(ran(split))
grd9 : :ard(dom(map)) = card(ran(split))

INITIALISATION/inv12/INV
MapPhase/inv5/INV
MapPhase/inv6/INV
MapPhase/inv10/INV
MapPhase/inv12/INV
MapPhase/act4/WD
```

Figure 6: Data Completeness property.

A set of proof obligations is then generated. If a proof obligation is not proven, then the designer uses the trace file generated from the transformation phase to locate the error in the BPMN specification and fix it.

5 VERIFICATION AND VALIDATION

Uses Case. The "Fireware" application is the solution to the "Denial-of-Service" DOS attack problem. Figure 5 illustrates the MapReduce workflow of the Fireware application, composed by three patterns: occurrence calculation pattern, sort pattern and filter pattern. The first pattern "Occurrence Calculation" processes a very large log file that contains all the IP addresses that have accessed a given URL to calculate the number of hits for each IP address. The list of key-value pairs produced by this pattern will be sorted in an array by the following pattern "Sort". The last pattern "Filter" will take the sorted array and apply a filter (maximum number of accesses allowed) on all key (IP address), value (number of accesses) pairs.

Figure 7 represents the architecture of the Event B project corresponds to the example in figure 5. Four machines linked by refinement are created: the first machine created is the machine named "MACHINE_APPLICATION" which contains only the information concerning the control flows of the "Fireware Application" participant. The second machine, "MachinePatron1", preserves the previous machine and contains the information concerning the control flows of the first sub-Process "Pattern calculation of occurrence" and its internal functioning. The third "MachinePatron2" and the fourth machine "Machine Patron3" contain information concerning the other two sub-processes "Sorting Pattern" and "Filtering Pattern". Each machine preserves the consistency of its previous one.

Verification and Validation. The most important interest of the proposed translation of MapReduce workflow into Event B is to allow the formal verification of safety properties. We are interested in verifying data quality properties(Zhang et al. 2020) to ensure the proper functioning and correctness of our MapReduce application and they are generated automatically by the transformation process.

- Data Completeness. It guarantees that all key-value pairs are processed without missing ones. For example, figure 6 illustrates the data completeness property between the two phases Split Data and Map Phase. It is represented in the form of an invariant (inv12) and a guard (grd9) in the machine "MachinePattern1". Ensure that the input data cardinality of the map task is equal to the output data cardinality of the split task.

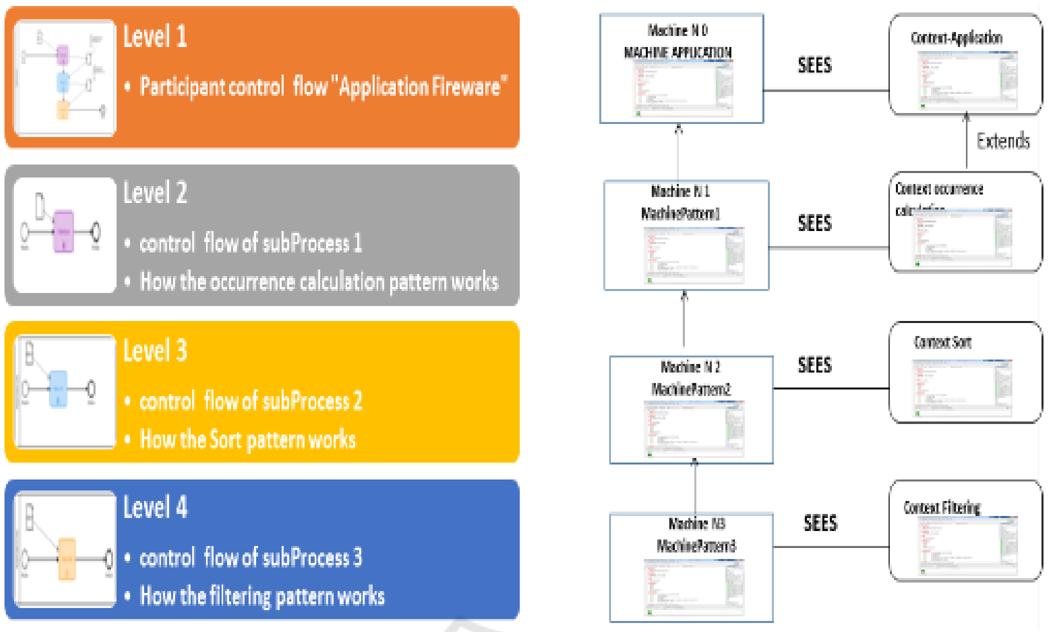
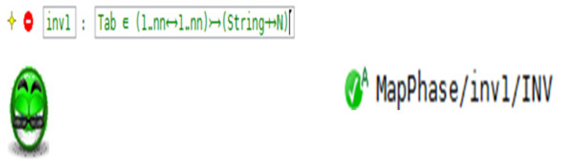

Level 1
• Participant control flow "Application Fireware"
Level 2
• control flow of subProcess 1
• How the occurrence calculation pattern works
Level 3
• control flow of subProcess 2
• How the Sort pattern works
Level 4
• control flow of subProcess 3
• How the filtering pattern works
Machine N 0
MACHINE APPLICATION
SEES
Context-Application
Extends
Machine N 1
MachinePattern1
SEES
Context occurrence calculation
Machine N 2
MachinePattern2
SEES
Context Sort
Machine N3
MachinePattern3
SEES
Context Filtering

Figure 7: The architecture of the RODIN project of the "firewire" application.

- Data non-conflict. It guarantees the absence of two key-value pairs with the same key and different values in any list of key-value pairs except if they cooperate. This property is necessary for applications for processing arrays where we try to avoid the problem of duplicate values as in our Sort pattern. The Map phase of the Sort pattern will process the data provided by the Reducer phase of the occurrence calculation pattern. It will store the key-value pairs in an array and therefore the result requires the absence of the duplications. To do so, the array must be a bijective function between its domain and its range. This property is represented in the invariant inv1 of our machine.


inv1 : Tab ∈ (1‥nn↔1‥nn)⤖(String↔N)

MapPhase/inv1/INV

Figure 8: Non-conflict property of the Sort pattern.

6 CONCLUSION

In this paper, a model-driven approach is presented to ensure the reliability of MapReduce applications. The user can easily use the Camunda graphical tool of BPMN to construct his MapReduce workflow by using a chosen set of MapReduce design patterns. The graphical model is then transformed automatically to an Event B project composed by a set of contexts and machines linked by refinement. The use of refinement in the proposed approach improve the readability and enhance the provability of the generated formal specification. Furthermore, a set of data driven properties such as data non-conflict and data completeness are automatically generated by the transformation process and then verified by the Rodin prover.
In the future, we plan to integrate more design patterns in our approach and study the possibility of combining Model-Checking and Theorem-Proving to benefit from the advantages of each technique for the verification of other types of properties such as the temporal property.

REFERENCES

Abrial, Jean-Raymond. 2010. *Modeling in Event-B: System and Software Engineering*. Edited by Cambridge University Press. 1 edition.

Bersani, Marcello M., Francesco Marconi, Damian A. Tamburri, Andrea Nodari, and Pooyan Jamshidi. 2019. "Verifying Big Data Topologies By-Design: A Semi-Automated Approach." *Journal of Big Data* 6 (1). https://doi.org/10.1186/s40537-019-0199-y.

Bessifi, Mayssa, Ahlem Ben Younes, and Leila Ben Ayed. 2022. "BPMN2EVENTB Supporting Transformation from BPMN2.0 to Event B Using Kermeta." In *Lecture Notes in Networks and Systems*, 286:247–55. https://doi.org/10.1007/978-981-16-4016-2_24.

Chiang, Dai Lun, Sheng Kuan Wang, Yu Ying Wang, Yi Nan Lin, Tsang Yen Hsieh, Cheng Ying Yang, Victor R.L. Shen, and Hung Wei Ho. 2021. "Modeling and Analysis of Hadoop MapReduce Systems for Big Data Using Petri Nets." *Applied Artificial Intelligence* 35 (1): 80–104. https://doi.org/10.1080/08839514.2020.18421 11.

Colas, Mathieu, Ingo Finck, Jerome Buvat, Roopa Nambiar, and Rishi Raj Singh. 2014. "Cracking the Data Conundrum : How Successful Companies Make Big Data Operational." *Capgemini Consulting*, 17. https://www.capgemini.com/consulting/wp-content/ uploads/sites/30/2017/07/big_data_pov_03-02-15.pdf %0Ahttps://www.capgemini-consulting.com/resource-file-access/resource/pdf/cracking_the_data_conundru m-big_data_pov_13-1-15_v2.pdf.

Correia, Anacleto, and Fernando Brito e Abreu. 2012. "Adding Preciseness to BPMN Models." *Procedia Technology* 5: 407–17. https://doi.org/10.1016/j.protcy. 2012.09.045.

Duan, Zhenhua, Cong Tian, Nan Zhang, Qian Ma, and Hongwei Du. 2019. "Index Set Expressions Can Represent Temporal Logic Formulas." *Theoretical Computer Science* 788 (November): 21–38. https://doi.org/10.1016/j.tcs.2018.11.030.

Hong, Jang Eui, and Doo Hwan Bae. 2000. "Software Modeling and Analysis Using a Hierarchical Object-Oriented Petri Net." *Information Sciences* 130 (1–4): 133–64. https://doi.org/10.1016/S0020-0255(00)00090-6.

Lim, Harold, Herodotos Herodotou, and Shivnath Babu. 2012. "Stubby:A Transformation-Based Optimizer for Mapreduce Workflows." *Proceedings of the VLDB Endowment* 5 (11): 1196–1207. https://doi.org/10. 14778/2350229.2350239.

Perez-Palacin, Diego, José Merseguer, José I. Requeno, M. Guerriero, Elisabetta Di Nitto, and D. A. Tamburri. 2019. "A UML Profile for the Design, Quality Assessment and Deployment of Data-Intensive Applications." *Software and Systems Modeling* 18 (6): 35 77–3614. https://doi.org/10.1007/s10270-019-00730-3

Rehan, Mohd, and Durgaprasad Gangodkar. 2015. "Hadoop , MapReduce and HDFS : A Developers Perspective." In *Procedia Computer Science*, 48:45–50. Elsevier. https://doi.org/10.1016/j.procs.2015.04.108.

Ruiz, M. Carmen, Javier Calleja, and Diego Cazorla. 2015. "Petri Nets Formalization of Map/Reduce Paradigm to Optimise the Performance-Cost Tradeoff." In *Proceedings - 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015*, 3:92–99. IEEE. https://doi.org/10.1109/Trustcom.2015.617.

S, athisha M, and K C Ravishankar. 2019. "Security and Privacy Issues in Big-Data Hadoop A Review." *International Journal of Computer Sciences and Engineering* 6 (11): 730–38. https://doi.org/10. 26438/ijcse/v6i11.730738.

Toman, Zinah Hussein, Lazhar Hamel, Sarah Hussein Toman, Mohamed Graiet, and Dalton Cézane Gomes Valadares. 2024. "Formal Verification for Security and Attacks in IoT Physical Layer." *Journal of Reliable Intelligent Environments* 10 (1): 73–91. https://doi.org/10.1007/s40860-023-00202-y.

Wang, Meng, Cong Tian, Nan Zhang, Zhenhua Duan, and Chenguang Yao. 2020. "Translating Xd-C Programs to MSVL Programs." *Theoretical Computer Science* 809: 430–65. https://doi.org/10.1016/j.tcs.2019.12.038.

Zeliu, Ding, Guo Deke, Chen Xi, and Chen Jin. 2019. "MapReduce Rationality Verification Based on Object Petri Net." *Journal of Systems Engineering and Electronics* 30 (5): 861–74. https://doi.org/10.21629/J SEE.2019.05.05.

Zhang, Nan, Meng Wang, Zhenhua Duan, and Cong Tian. 2020. "Verifying Properties of MapReduce-Based Big Data Processing." *IEEE Transactions on Reliability*, 1–18. https://doi.org/10.1109/tr.2020.2999441.

SCIENCE AND TECHNOLOGY PUBLICATIONS