International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 1, February 2020, pp. 377~386
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i1.pp377-386

Analysis and implementation of the impact of change: application to heterogeneity algorithms in enterprise architecture

Jihane Lakhrouit[1], Karim Baïna[2]
[1]Department of Computer Science, Higher Institute of Engineering and Business, ISGA Marrakech, Morocco
[1,2]University Mohammed V, ENSIAS Alqualsadi research team on Enterprise Architecture, Morocco

Article Info

Article history:

Received Jan 16, 2019
Revised Jul 31, 2019
Accepted Aug 29, 2019

Keywords:

Analysis of EA
Complexity
EA patterns
Enterprise architecture EA
Heterogeneity

ABSTRACT

Measurements play an important role in many scientific fields in general and in the analysis of enterprise architecture in particular. In software engineering, the measures are used to control the quality of the software product and better manage development projects to control the cost of production. In this article we proposed firstly models and measures to evaluate and analyze the complexity of the enterprise architecture and especially the heterogeneity of components and relationships, and secondely we developed a model to automatically detect the change of measures and their impact on enterprise architecture.

Corresponding Author:

Jihane Lakhrouit,
Higher Institute of Engineering and Business, ISGA Marrakech,
Department of Computer Science, University Mohammed V, ENSIAS,
Morocco.
Email: jihane.lakhrouit@gmail.com

1. INTRODUCTION

Today, many organisations are concerned with how to successfully transition to organisations utilising information technology to its fullest strategic extent. It has become widely recognised that an organisation's enterprise architecture plays a key role in the transition and many organisations are now investing significant amounts of resources into developing or improving their enterprise architecture [1]. The enterprise architecture (EA) is the organizing logic for business processes and IT infrastructure, reflecting the integration and standardization requirements of the company’s operation model, to analyze the result of enterprise architecture we present in this paper, a complete methodology for analyzing the heterogeneity of enterprise architecture. Our objective is to propose an evaluating methodology for guiding designers and architects in evaluating and improving the EA models. Furthermore, our enterprise architecture patterns system will be used for an automated support to manage the evaluation of enterprise architecture complexity

The goal of this paper is to (1) present the enterprise architecture component regarding agility and complexity measurement, (2) identify and apply the heterogeneity metrics to enterprise architecture components and relationships (4) Detect changes in an enterprise architecture and update relevant metrics. The paper is structured as follows: the second et alion describes the state of the art of our research, the third et alion presents our proposed approach and presents some of our results, the fourth et alion presents the pototype of our contribution and finally, the last et alion is dedicated to conclude our paper.

2. STATE OF THE ART

Enterprise architecture (EA) has in recent years tremendously increased across industries, many organizations continue to encounter challenges which affect the development, implementation, and practice [2]. Enterprise Architecture (EA) is a strategy to attain alignment between an enterprise’s business and Information Technology (IT) to increase the competitiveness of an enterprise [3]. Among the success factors of this alignment is the study of complexity.

Complexity is considered one of the most critical issues to deal with because of the constraints and difficulties that surround it [4], many companies seem to consider it as a general problematic source, it is held responsible for the rise in coordination efforts [5], operating costs, and also increased effort to make changes, which significantly hinders the agility and alignment of the information system [6, 7]. The Cambridge Dictionary defines complexity as “the state of having many parts and being difficult to understand or find an answer to”. Much of the existing architecture research endorses this view, by relating complexity to the number of components or elements, their relationship, and totheir variation/variety, and heterogeneity [8-11] adds that the total complexity of an EA must take into account complexity within each domain, as well as the complexity of the interrelations between domains [12]: According to Davis and LeBlanc [13] the complexity of application architecture is “number of its components or elements, kind or type of elements and structure of the relationship between elements”. On the infrastructure architecture level defined complexity as “The complexity can be defined here as the dramatic increase in the number and heterogeneity of included components, relations, and their dynamic and unexpected interactions in IT solutions” [14], another definition proposed by [15] covers all aspects of complexity “The complexity can be defined on the basis of the number and variety of components and interactions plus the rate of change of these”. From the different definitions cited we can notice that the complexity is a fuzzy term, because different stakeholders have generally different views and conceptions of complexity term. From these different definitions we will clarify the dimensions of complexity and proposed a global definition: “The complexity of architecture is the description of its structure and quantification of the numbers and heterogeneity of components and relations between them over the time” [16]. In this paper, we will discuss the dimension of enterprise architecture heterogeneity (components and relations) and also the rate and impact of change of heterogeneity dimension.

During the analysis of the identified contributions wich discussed enterprise architecture evaluation complexity only few methods were presented to quantify complexity and the existing methods merely cover parts of an EA, not the EA as a whole. Often the application is so specific that it is not possible to transfer the method to other dimensions of an EA. In the paper [17] it discussed the metrics for EAs and application landscapes are introduced as decision support techniques based on analysis of structural dependencies. The approach emphasizes on operational risk, failure propagation and availability, based on a practitioner survey. In order to explicate the structural dependencies analyzed in the paper, an information model with derived attributes is used, along with Bayesian calculation formalism. An EA level application example is also given in the paper [17] with visual analysis of ex post information about failure propagation to compare different project proposals for the evolution of the application landscape. Thus, the project portfolio management process is supported. Lagerström et al. [18] proposed to use an approach pervasive in the software architecture discipline— Design Structure Matrix—to visualize the hidden structure of an AL and thereby identify spots of increased complexity. Schuetz et al. [19] introduce a metric to quantify the structural complexity of an IT landscape, which is also applicable to application landscape. The proposed approach of Schutz [19] revolves around the conceptualization of the complexity of EA by adopting the concept of the system to the context of EA. This approach presented a holistic conceptualization of complexity but don’t apply it in the different layers of EA. After define and clarify the dimensions of complexity we present our contribution to modelise and evaluate EA complexity.

3. OUR PROPOSAL PATTERNS FOR MODELLING

This et alion presents the information patterns for the analysis of the enterprise architecture. We define firstly the patterns to analyze and implementing enterprise architecture heterogeneity algorithms and secondly we detail our approach to modelize the impact of the changing algorithms.

3.1. Definition and conceptual foundation

Heterogeneity is defined as the diversity of elements or relationships of a system according to its characteristics [20]. More precisely, in computer science, the heterogeneity of a computer landscape is a statistical property that presents the diversity of the types of elements that compose it [17, 21] taking as an example the heterogeneity of database management systems (DBMS). This heterogeneity can be understood as a frequency distribution [22, 23] and can be expressed in graphical form as shown in the Figure 1.

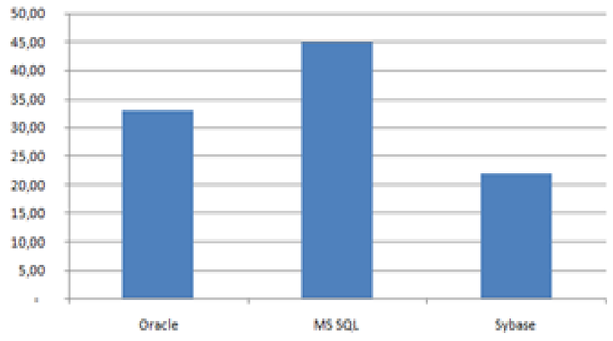


Figure 1. The number of instances per DBMS

In the literature the most widely used method for measuring heterogeneity is the use of concentration measurements, which is entropy measure $H = EM = -\sum_{i=1}^{n} p_i \ln(p_i)$ [23, 24].

3.2. Analyzing enterprise architecture heterogeneity

Based on the information pattern I-50 presented on the paper [25] we present three types of concepts in which we apply the measure of entropy. Concept 1 represents only the heterogeneity of a single component of the enterprise architecture, concept 2 represents the relationship between two components and calculates heterogeneity with respect the relation and the concept 3 is an exceptional case from concept 2 it presents a relationship path that connects several components. These concepts are summarized in the Table 1. The I- pattern I-52 presents the measurements detailed in the Table 1. The measurements are illustrated and numbered from 1 to 8 in the diagram (Figure 2).

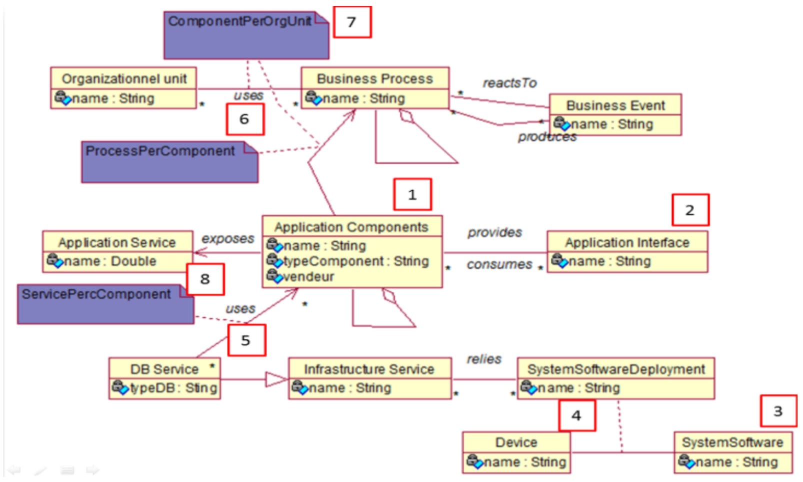


Figure 2. The I-Pattern diagram "Analysis of Heterogeneity" I-52

Table 1. The application of heterogeneity to the threeconcepts of the heterogeneity measurements

Concept Type	Concept of Heterogeneity	Number of instances	The Heterogeneity of the Concept
Type 1	Application Components	Number of Application Components	Concentrations of applications by vendor or type (developed, purchased and adapted, purchased).
	Application Interface	Number of Interfaces	Concentrations of the types of interfaces.
	Computer	Number of Computers	Computer Concentrations by Type
	Operating System	Number of Operating System	Operating System Concentrations by Type
	Database	Number of databases	Database Concentrations by Type
Type 2/3	Implemented Processes	Number of Implemented Processes	Concentration of implemented processes by component
	Using application components	Number of components used by organizational units	Concentration of processes by organizational unit.
	Using Databases	Number of database instances used.	Concentration of databases by component

3.3. Implementation of analysis algorithms

To propose an evolutionary implementation we must consider several constraints: 1- these algorithms can evolve over time, 2- we can have several versions of the same algorithm during the life cycle of our system and each version can represent an adaptation or an optimization of the old version, 3- we also want to isolate the algorithms compared to others to facilitate their use their implementation and maintenance. These cited constraints were managed and resolved by the "Strategy" design pattern; for that we will adapt the design pattern "Strategy" and apply it to our context. The Figure 3 shows the application of the design pattern to our context. We create a “StrategyInterface’’ interface, we add an “applyAlgorithm” method that will be the method that applies our strategy or in other words that implements our algorithm. Concrete classes created implement this interface to encapsulate the algorithms and to redefine the “applyAlgorithm” method for implementing the algorithm of each class. In our contribution we proposed an algorithm hierarchy using the notion of "Abstract Class", we represent two large families of algorithms; the heterogeneity algorithms "AlgorithmeHeterogeneite" divided into two subtypes; type 1 algorithms and type 2 algorithms The Figure 4 shows an example of implementation and use of the database concentration algorithm.

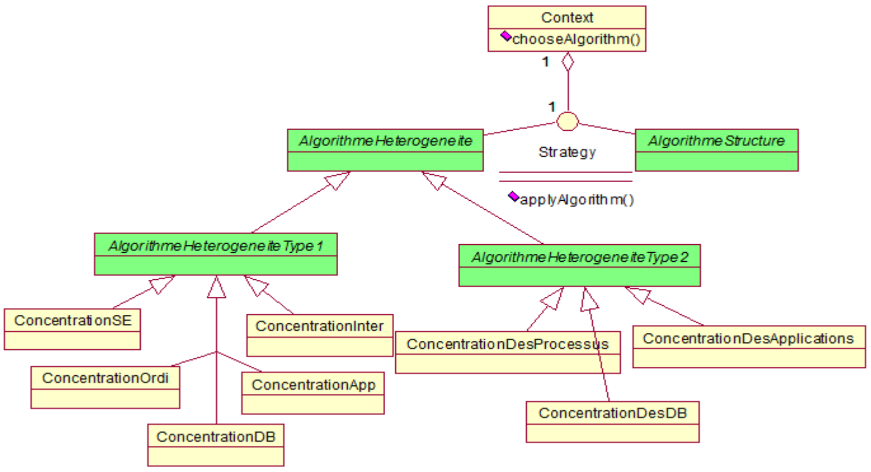
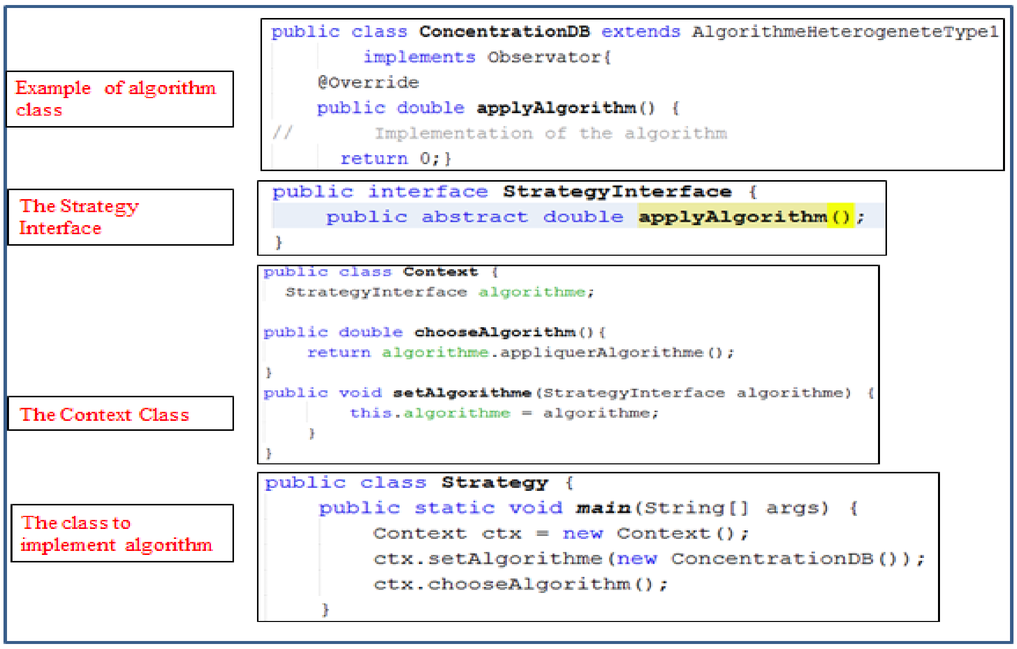


Figure 3. The implementation of strategy design pattern in our context



```
public class ConcentrationDB extends AlgorithmeHeterogeneteType1
        implements Observator{
    @Override
    public double applyAlgorithm() {
//          Implementation of the algorithm
      return 0;}
```

```
public interface StrategyInterface {
    public abstract double applyAlgorithm();
}
```

```
public class Context {
  StrategyInterface algorithme;

public double chooseAlgorithm(){
    return algorithme.appliquerAlgorithme();
}
public void setAlgorithme(StrategyInterface algorithme) {
        this.algorithme = algorithme;
    }
}
```

```
public class Strategy {
    public static void main(String[] args) {
        Context ctx = new Context();
        ctx.setAlgorithme(new ConcentrationDB());
        ctx.chooseAlgorithm();
    }
```

Figure 4. An example of the implementation of Concetration databases algorithm

```
Name: calculation of process concentration by component
Variables: BS: all business processes
CP: the application components
 Map instances = map <String componentType, Integer processNumber>
Double sum
Double percentage
Integer Comp
Double heterogeneity
Create a map = instance: its key is a String for the application components and an integer for the number of processes
For all cp in CP do
For all r in cp.relations do
   If (r.target = bs) then
count = count + 1
         If instances contains componentType = cp.name
For any instance in the instances map
If (instance.composingType == cp.name)
Increment the number numberProcess by 1
End if
endfor
          If not
Add a new entry in the map with the key cp.name and value 1
          End if
    If not
Do nothing and move on to the next relationship
   End if
End For
End For
For any instance in the instances map
// Divide instance.numberProcess by count
Double percentage = instance. numberProcess / comp
sum = sum + percentage * ln (percentage)
endfor
heterogenity = -som
return heterogenite
```

3.4. Analysis the impact of change

Among the dimensions of complexity presented in the et alion 1, we have specified the impact of change as an important dimension to consider; in this et alion we will propose an implementation to resolve this need. The impact of managed change in our contribution is to automatically update the new measures and to progressively follow the changes of our proposed system proposed in the I-Pattern I-52 "Heterogeneity of Enterprise Architecture". In this et alion we will propose an implementation that detects the change of the considered components and reflects this change at the level of the measurement algorithms. To handle these constraints we propose to use the observer design patten. This pattern presents a solution to send a notification to modules that play the role of observers. In the event of notification, the observers take the appropriate action according to the information that arrives from the modules they observe (the "observables").

The diagram of the Observer pattern illustrated in the Figure 5 presents the proposed solution, it defines two interfaces and two classes: The **Observer interface** will be implemented by any class that wants to be an observer. This is the case of the ObservatorConcret class which implements the Observable method, this method will be called during a state change of the observed class. There is also an **Observable interface** that will be implemented by the classes that we want to observe. The **ObservableConcret** class implements this interface, which allows it to keep observers and informed by notifying them. Each ObservableConcret class has an attribute (or several) that we want to observe and a list of observers. The state is an attribute whose observers wish to follow the evolution of its values. The list of observers is the list of observers who are listening. The **ObservableConcret** class in our context is the EAModel class, it represents our ArchiMate models. This class will contain two elements: components and relationships. The EAModel class has the states that we want to observe, which are all the nodes and relationships of the enterprise architecture landscape.

The EAModel class also contains all observers who will receive notifications on each change. The ObserversConcret who are listening are the implementation classes of the analysis algorithms. If a component or relationship is added, deleted, or modified, the observers concerned with this model update are refreshed automatically. In our model the concrete observers are the algorithms of heterogeneity as shown in Figure 6.

```
Name: calculation of the heterogeneity of operating systems and computers
Variables: SSD: All Instances of Operating Systems Deployed SystemSoftwareDeployment
Map instancesSE = map <String instanceType, Integer numberInstance>
Map instancesComputer = map <String instanceType, Integer numberInstance>
Double sum, sumDorDI
Double percentageSE, percentage ORDI
Double heterogeneity ORDI, heterogenite
Integer numberInstanceSE, numberInstanceORDI

Create a map = instanceSE that has a String for the OS type and an integer for the number of instances
Create a map = instanceComputer that has a String for the computer type and an integer for the number of instances
For all ssd in SSD
If instanceSE contains instanceType = ssd.systemSoftware
numberInstanceSE = instanceSE.get (ssd.systemSoftware)
Increment the number numberInstanceSE by 1
instanceSE.get (ssd.systemSoftware) .SetValue (nombreInstanceSE)
If not
Add a new entry in the instanceSE map as ssd.systemSoftware key and  value 1
InstanceSE.add (ssd.systemSoftware, 1)
End if

If the computer instance contains instanceType = ssd.device
numberInstanceORDI = instanceComputer.get (ssd.device)
Increment the numberComputer instance by 1
instanceSE.get (ssd.device) .SetValue (nombreInstanceORDI)
If not
Add a new entry in the computer instance instance as ssd.device key and the value 1
instanceOrdinateur.add (ssd.device, 1)
End if
End For

For i ranging from 0 to N = SSD.size ()
 // divide numberInstance by N
Double percentage = instancesSE.get (i) .getValue () / N
sum = sum + percentage * log (percentage)
 endfor
heterogeniteSE = -som
 sum = 0

For i from 0 to N = SSD.size ()
 // divide numberInstance by N
Double percentage = computerInstance.get (i) .getValue () / N
sum = sum + percentage * log (percentage)
 endfor
heterogeniteORDI = -som
```

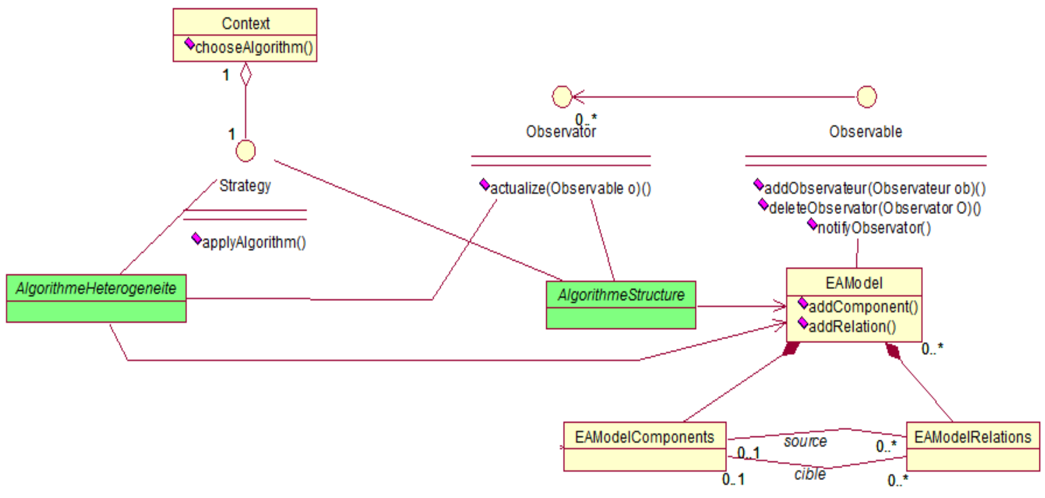
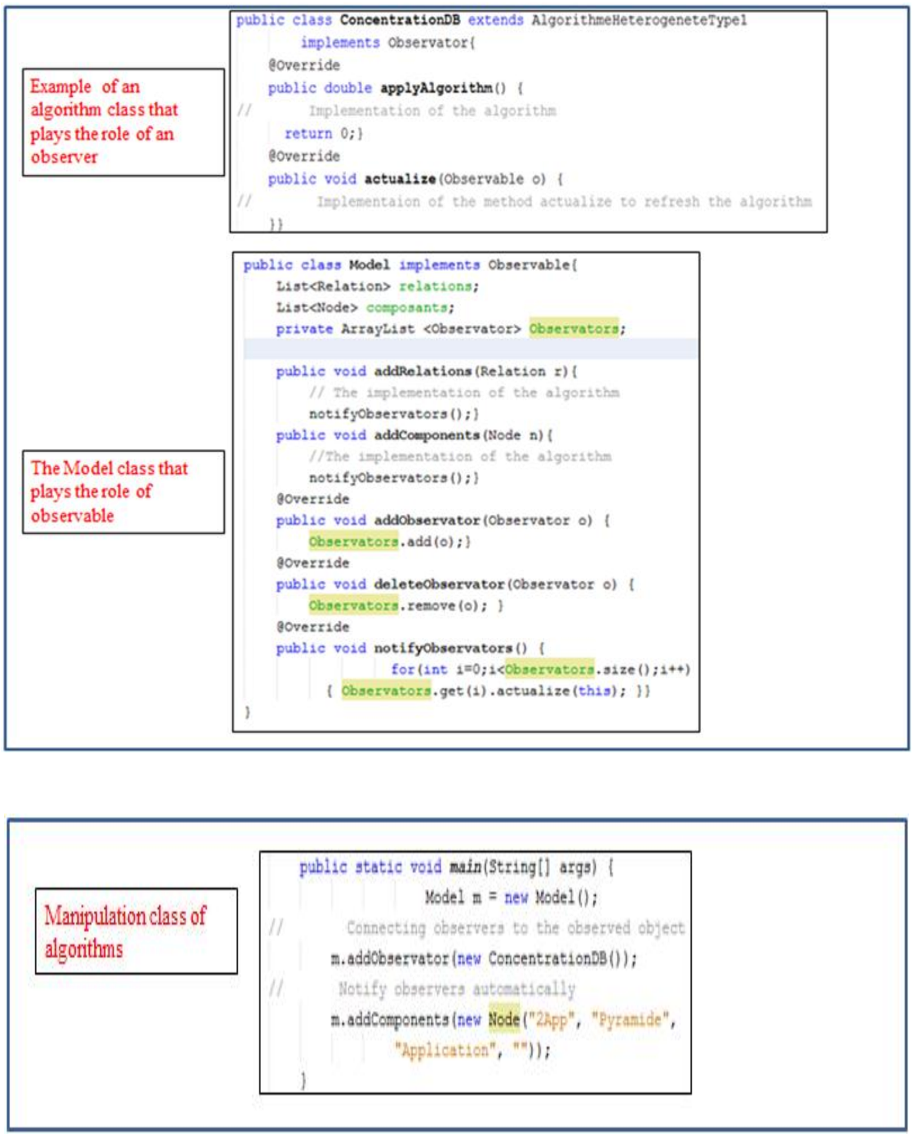


Figure 5. The implementation of observer design pattern in our context


Example of an algorithm class that plays the role of an observer
The Model class that plays the role of observable
Manipulation class of algorithms

```java
public class ConcentrationDB extends AlgorithmeHeterogeneteType1
        implements Observator{
    @Override
    public double applyAlgorithm() {
//          Implementation of the algorithm
      return 0;}
    @Override
    public void actualize(Observable o) {
//           Implementaion of the method actualize to refresh the algorithm
    }}
```

```java
public class Model implements Observable{
    List<Relation> relations;
    List<Node> composants;
    private ArrayList <Observator> Observators;

    public void addRelations(Relation r){
        // The implementation of the algorithm
        notifyObservators();}
    public void addComponents(Node n){
        //The implementation of the algorithm
        notifyObservators();}
    @Override
    public void addObservator(Observator o) {
        Observators.add(o);}
    @Override
    public void deleteObservator(Observator o) {
        Observators.remove(o); }
    @Override
    public void notifyObservators() {
                for(int i=0;i<Observators.size();i++)
          { Observators.get(i).actualize(this); }}
}
```

```java
    public static void main(String[] args) {
                    Model m = new Model();
//        Connecting observers to the observed object
        m.addObservator(new ConcentrationDB());
//       Notify observers automatically
        m.addComponents(new Node("2App", "Pyramide",
                "Application", ""));
    }
```

Figure 6. An example of the implementation of observer design pattern

4. PROTOTYPE

The application architecture is divided into three layers: an information management or backup layer that stores data from a model or from existing source files in a data warehouse, a reporting layer that presents the results as shwon in Figure 7. Heterogeneity measures in graphical form and an interaction layer that offers the possibility of modeling the desired points of view.

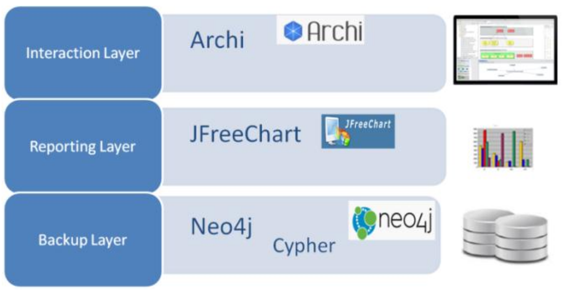


Figure 7. The three layers of prototype

The interaction layer represents the applications that will allow decision makers to model the views of the enterprise architecture and enrich it with existing data. The modeling editor is as shown in Figure 8. The illustrated tool represents the first step which is the modeling of the enterprise architecture by graphically describing the elements and existing relations, it is an ArchiMate point of view modeled by the Archi interface. It consists of an element set of each layer. The description of the AE is stored in two Comma-separated values CSV files. To manage this metadata, we have developed a desktop application java, illustrated in the Figure 9, which allows us to manage this metadata, to apply the heterogeneity measurement algorithms and to visualize the output graphs.

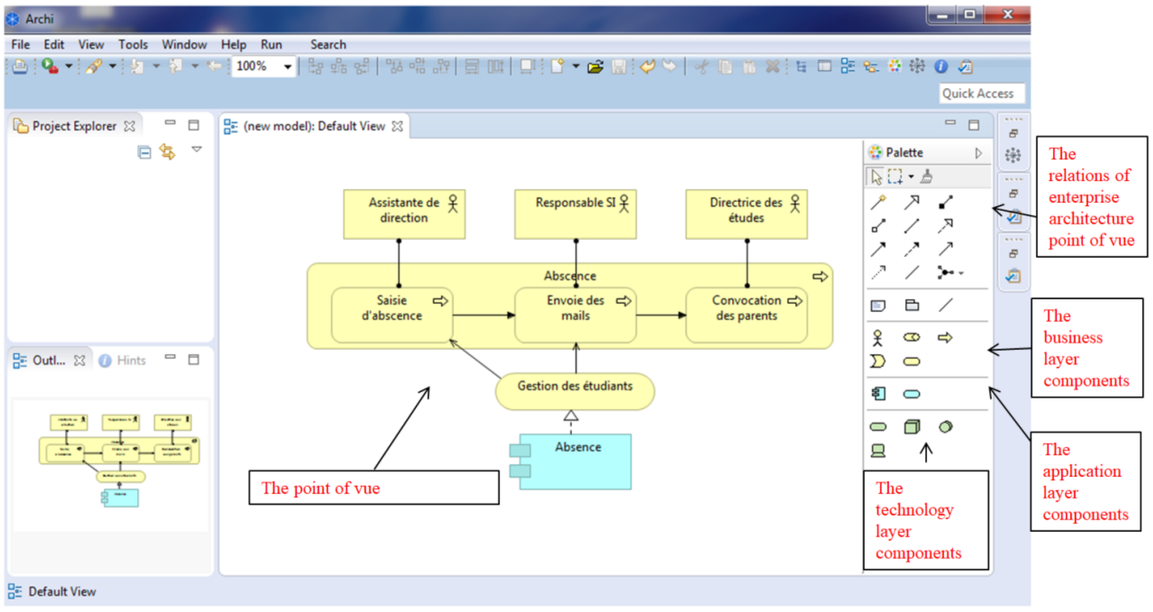
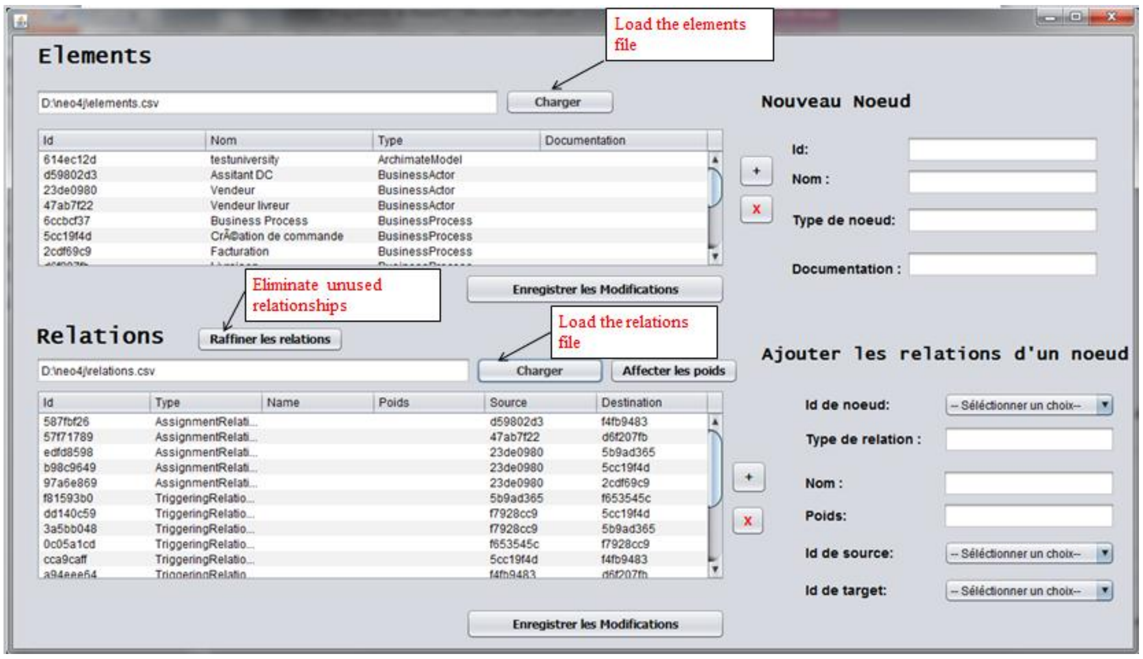


Figure 8. The modeling interface



Elements

D:\neo4j\elements.csv

Id	Nom	Type	Documentation
614ec12d	testuniversity	ArchimateModel	
d59802d3	Assitant DC	BusinessActor	
23de0980	Vendeur	BusinessActor	
47ab7f22	Vendeur livreur	BusinessActor	
6ccbcf37	Business Process	BusinessProcess	
5cc19f4d	CrÃ©ation de commande	BusinessProcess	
2cdf69c9	Facturation	BusinessProcess	

Relations

D:\neo4j\relations.csv

Id	Type	Name	Poids	Source	Destination
587fbf26	AssignmentRelati...			d59802d3	f4fb9483
57f71789	AssignmentRelati...			47ab7f22	d6f207fb
edfd8598	AssignmentRelati...			23de0980	5b9ad365
b98c9649	AssignmentRelati...			23de0980	5cc19f4d
97a6e869	AssignmentRelati...			23de0980	2cdf69c9
f81593b0	TriggeringRelatio...			5b9ad365	f653545c
dd140c59	TriggeringRelatio...			f7928cc9	5cc19f4d
3a5bb048	TriggeringRelatio...			f7928cc9	5b9ad365
0c05a1cd	TriggeringRelatio...			f653545c	f7928cc9
cca9caff	TriggeringRelatio...			5cc19f4d	f4fb9483
a94eee64	TriggeringRelatio...			f4fb9483	d6f207fb

Nouveau Noeud

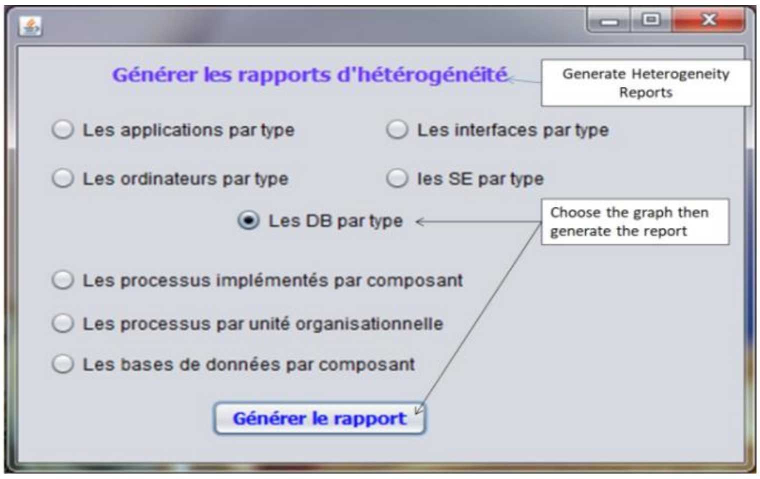
Id:	
Nom :	
Type de noeud:	
Documentation :	

Ajouter les relations d'un noeud

Id de noeud:	-- Sélectionner un choix--
Type de relation :	
Nom :	
Poids:	
Id de source:	-- Sélectionner un choix--
Id de target:	-- Sélectionner un choix--

Figure 9. The Meta data management of EA components and relationships

To manage this metadata, we have developed a java desktop application, illustrated in Figure 10, that allows us to load relationships and components from csv files, view them and make changes if necessary. Figure 11 show the report generated for the distribution of databases instances.



Générer les rapports d'hétérogénéité

Generate Heterogeneity Reports

Les applications par type

Les interfaces par type

Les ordinateurs par type

les SE par type

[✓ Les DB par type]

Choose the graph then generate the report

Les processus implémentés par composant

Les processus par unité organisationnelle

Les bases de données par composant

Générer le rapport

Figure 10. The interface to generate the heterogeneity graphs

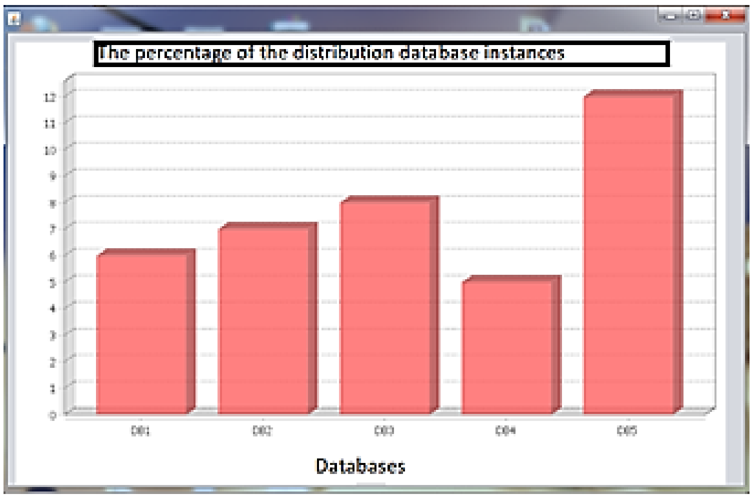


Figure 11. The report generated for the distribution of databases instances

4 CONCLUSION

Enterprise Architecture (AE) is a cross-cutting discipline that deals with the process, models, tools for describing organizations and building their IS. It also helps to plan the possible changes at the organizational level and the architecture level. As a result, different approaches have been employed to ascertain the challenges, yet they persist. Thus, the objective of this paper is to propose an evaluating methodology for guiding designers and architects in evaluating and improving the EA models and especially the impact of the change of the different components at the level of the complexity measures.

REFERENCES

[1] M. Hall and Tideman, "Measures of Concentration," *Journal of American Statistical Society*, vol. 62, pp. 162-168, 1967

[2] T. Iyamu, "Understanding the complexities of enterprise architecture through structuration theory," *Journal of Computer Information Systems*, vol. 59, pp. 287-295, 2019.

[3] Rouhani, B. D., Ahmad, R. B., Nikpay, F., & Mohamaddoust, R. (2019). CRITICAL SUCCESS FACTOR MODEL FOR ENTERPRISE ARCHITECTURE IMPLEMENTATION. Malaysian Journal of Computer Science, 32(2), 133-148.

[4] C. Lucke, S. Krell, and U. Lechner, "Critical issues in enterprise architecting: A literature review," In Americas Conference on Information Systems, Lima, 2010.

[5] C. B. Daniels, and W. J. LaMarsh II, "Complexity as a cause of failure in information technology project management," In IEEE International Conference on System of Systems Engineering, 1– 7, San Antonio, TX, 2007.

[6] K. Wehling, D. Wille, C. Seidl, and I. Schaefer, "Decision support for reducing unnecessary IT complexity of application architectures," In 2017 IEEE International Conference on Software Architecture Workshops, 161–168, 2017.

[7] C. Schmidt, "Business architecture quantified: How to measures business complexity," In D. Simon and C. Schmidt (Eds.), Business Architecture Management. Springer, 243–268, 2015.

[8] J. S. Davis, and R. J. LeBlanc, "A study of the applicability of complexity measures,"IEEE Transac- tion on Software Engineering,14(9), 1366–1372, 1988.

[9] R. L. Flood, and E. R. Carson, "Dealing with complexity: An introduction to the theory and applica- tion of systems science," In Dealing with Complexity: An Introduction to the Theory and Application of System Science, 23–38, Springer, 1993.

[10] W. Kinsner, W., "Complexity and its measures in cognitive and other complex systems," In 7th IEEE International Conference on Cognitive Informatics, 13–29, Stanford, 2008

[11] A. Schütz, T. Widjaja, and J. Kaiser, "Complexity in enterprise architectures – Conceptualization and introduction of a measure from a system theoretic perspective" In Proceedings of the 21stEuropean Conference on Information Systems, Utrecht, Netherlands, 2013.

[12] Iacob, M. E., Monteban, J., van Sinderen, M., Hegeman, E., & Bitaraf, K. (2018). *Measuring Enterprise Architecture Complexity. 2018 IEEE 22nd International Enterprise Distributed Object Computing Workshop (EDOCW).* doi:10.1109/edocw.2018.00026.

[13] J. S. Davis and R. J. LeBlanc, "A Study of the Applicability of Complexity Measures," *IEEE Transactions on Software Engineering*, vol. 14, pp. 1366.

[14] Y. Y. Yusuf and E. O. Adeleye, "A comparative study of lean and agile manufacturing with related survey of current practices in the UK," *International Journal of Production Research*, vol. 40, pp. 4545-4562, 2002.

[15] S. L. Schneberger and E. R. McLean, "The Complexity Cross: Implications for Practice," Communications of the ACM, vol. 46, pp. 216-225, 2003.

[16] J. Lakhrouit and K. Baïna, "A pattern based methodology for analyzing enterprise architecture landscape," International Journal of Computer Science Issues (IJCSI), vol. 13, pp. 15, 2016.

[17] Widjaja T., et al., "Heterogeneity in IT landscapes and monopoly power of firms: a model to quantify heterogeneity," The international conference on information systems (ICIS), Orlando, FL, 2012.

[18] Lagerstrom R., et al., "Visualizing and measuring enterprise application architecture: an exploratory telecom case," School working paper 13-103. Harvard Business, Cambridge, MA, 2013.

[19] Schutz, et al., "Complexity in enterprise architectures—conceptualization and introduction of a measure from a system theoretic perspective," The European conference on information systems (ECIS), Utrecht, The Netherlands, 2013.

[20] J. S. Davis and R. J. LeBlanc, "A Study of the Applicability of Complexity Measures," IEEE Transactions on Software Engineering, vol. 14, pp. 1366-1372, 1988.

[21] C. B. Garrison and A. S. Paulson, "An Entropy Measure of the Geographic Concentration of Economic Activity," Economic Geography, vol. 49, pp. 319-324, 1973.

[22] J. E. Kwoka Jr, "The Herfindahl Index in Theory and Practice," Antitrust Bull., vol. 30, pp. 915-947, 1985.

[23] F. M. Gollop and J. L. Monahan, "A Generalized Index of Diversification: Trends in U.S. Manufactoring," *The Review of Economics and Statistics*, vol. 73, pp. 318-330, 1991.

[24] A. P. Jacquemin and C. H. Berry, "Entropy Measure of Diversification and Corporate Growth," *The Journal of Industrial Economics*, vol. 27, pp. 359-369, 1979.

[25] J. Lakhrouit and K. Baïna, "A pattern based methodology for analyzing enterprise architecture landscape," *International Journal of Computer Science Issues (IJCSI)*, vol. 13, pp. 15, 2016.