Evaluation of Data Warehouse Systems by Models Comparison

Isaac Nkongolo Mbala and John Andrew van der Poll

***Abstract*—** Often in software engineering more than one solution to a complex problem is designed, thereafter selecting only the most appropriate one. A similar technique is used in requirements engineering – a key is to establish a foundation for assessing and choosing one model above others. This paper suggests a methodology through a case study around a data mart, to evaluate and compare Star and Snowflake models of data warehouse systems using a same set of requirements and from these furnish a means to choose the model deemed the more appropriate one. On the strength of the case study, the methodology suggests the elicitation of aspects relevant for the evaluation and comparison of the two models portrayed in Star and Snowflake schemas.

***Keywords*—** Case Study, Data Warehouse Design Models, Methodology, Requirements Definition, Snowflake Model, Star Model, UML Class Diagram.

I. Introduction

Critical systems such as data warehouse (DW) systems require a high level of software reliability and precision. In line with this purpose, the software development team ought to rely on alternatives to guide them towards a best practice solution.

A DW is defined as an integrated, subject-oriented and non-volatile collection of data that aids decision makers with strategic information [1]. DW system development differs from traditional systems embedding operational data bases since it's based on the support of the decision-making process (cf. Business Intelligence) of the enterprise [2]-[5].

The development of DW systems requires a set of phases to be performed, namely, requirements analysis, conceptual-, logical- and physical phases [6] [7] [4]. Among all these development phases, the requirements analysis and conceptual design are the two principal phases in the design process of DW systems [4]. Following this view, [8] contends the design process to be the most important phase in the successful construction of DW systems.

For DW systems, the conceptual design is the phase intended to yield the structural view of the system that is presented under the multidimensional form. The multidimensional model design is realized through an analysis of the business needs and business objectives. It consists of entities related among them such as facts, measures, dimensions and hierarchies [7] (refer Section II).

A DW is a complex structure [9] [10]; the cost of correcting a fault in the course of, or after the development, has been found to be costly compared to getting the same fault corrected earlier on. Therefore, much effort is put into correct design models to meet stakeholders' and decision makers' needs and expectations.

The Unified Modelling Language (UML) has been broadly accepted as a de facto standard for software design using semi-formal notations, yet often lacking precision or correctness [11]-[13]. In software development, several solutions can be considered for a same system and the assessment and selection of options are non-trivial [14] [15].

Selecting the most appropriate model from a set of options requires the approach to start with the same set of requirements for each option. The literature addresses cases of models being evaluated and compared [16] [17] and there are work on the design of DW systems using UML as an Object-Oriented Conceptual model [18]-[20].

Research has been conducted on the evaluation and comparison of DW Star and Snowflake models, stemming from UML variants, at the logical level using a multidimensional model. To the best of the knowledge of the authors, no work on comparing Star and Snowflake models at the conceptual level has been done.

This paper presents through a case study around a data mart a methodology for evaluating and comparing two main DW design models, namely, *Star* and *Snowflake*. We compare these design models using the same set of requirements expressed by stakeholders and decision makers.

The remainder of the paper is as follows: the definition of the research questions (RQs) below, Section II presents some design models of data warehouse systems. A framework to be used from previous work with all stages from the requirements definition to the comparison stage is illustrated in Section III. Section IV introduces the data-mart case study by defining the business requirements and business objective in Section IV.A followed by the generation of class diagrams for the Star and Snowflake models constructed from the requirements definition in sections IV.B and IV.C respectively. The results of comparisons appear in section IV.D and recommendations in Section V. Conclusions and directions for future work are given in Section VI.

A. Research Questions (RQs)

This paper aims to find answers to the following:

Manuscript received October 20, 2020. The University of South Africa (Unisa) Funding supported this work. Isaac Nkongolo Mbala is an MSc Candidate in the School of Computing, Unisa, Florida, South Africa.

John Andrew van der Poll is a Research Professor at the Graduate School of Business Leadership (SBL), Unisa, Midrand, South Africa.

RQ1: What are the main models used for the design of a data warehouse system?

RQ2: What may be the most suitable model for the design of a data warehouse system?

Next, we briefly address the data warehouse systems design models as mentioned in the RQs.

II. Data Warehouse (DW) Systems Design Models

A DW system is defined as the linking of a number of operational databases with (business) intelligence (decision-making) added to the resultant structure. An important substructure in a DW is the data mart; it is viewed as a subset of a DW (cf. a view in a database) often considered as one of the operational databases in the DW. Owing to its meta-level characteristic, the design of DW systems is rather different from the design of transactional systems, which supplies data to the warehouse [4].

Roughly, four models exist for the design of data warehouses. These are the star model, snowflake model, galaxy model and fact constellation model all aimed at representing the data warehouse system with emphasis on data structures at the logical level.

In this paper, we focus on the *star* and *snowflake* models for the design of DW systems by representing the two models at the conceptual level. These two models are considered the fundamental ones. Other models, namely, the galaxy- and fact constellation models are very similar – they combine several star models or snowflake models in a single diagram where different fact tables share some of the dimension tables and fact tables are also linked [21]. Examples of the galaxy and fact-constellation models appear in Fig. 1 and 2 respectively.

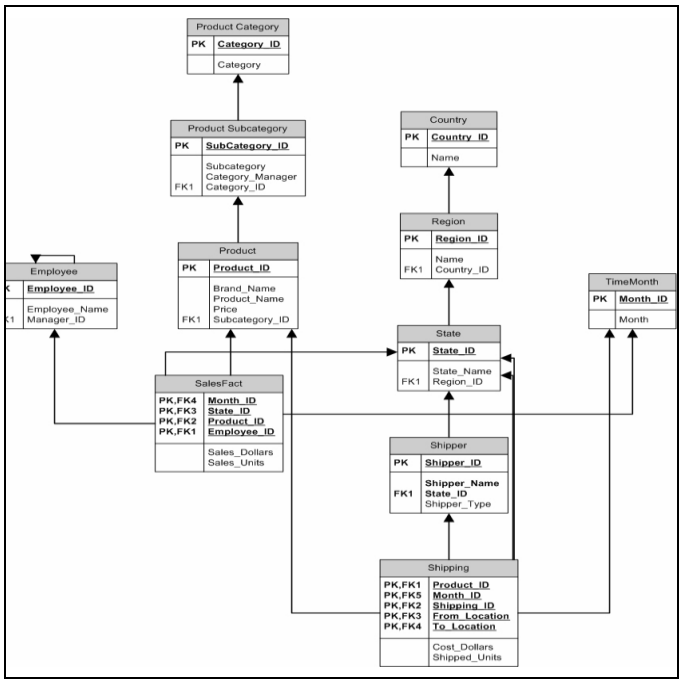

Fig. 1: Galaxy model [21]

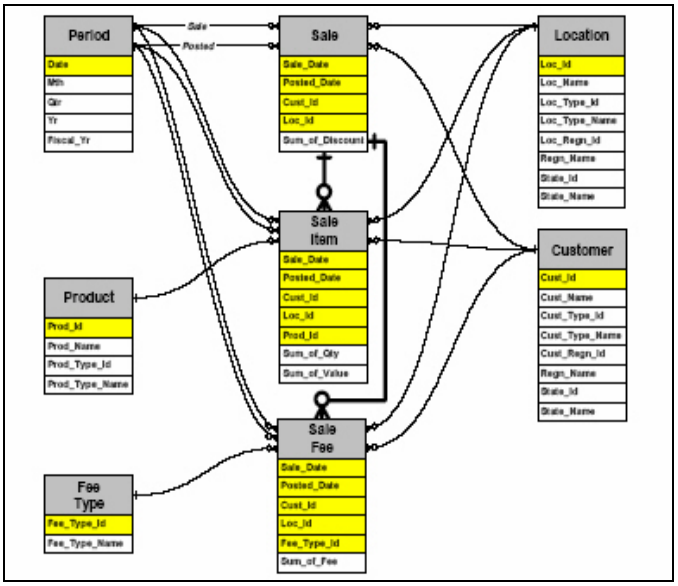

Fig. 2: Fact constellation model [21]

Further consideration of the galaxy- and fact-constellation models is beyond the scope of this paper.

A. Star Schema

A star is a relational database model utilized for containing measures and dimensions in a data mart [22]. Measures are numerical attributes stored in the fact table and dimensions are textual attributes maintained in dimension tables. A fact table is the subject analysis in the decision-making process and dimensions are axes of analysis [23]. A fact table is related to every dimension table. It is called "star" since the representation shows the fact table being surrounded by a number of dimension tables. Fig. 3 depicts a star model with 1 fact and 4 dimensions.

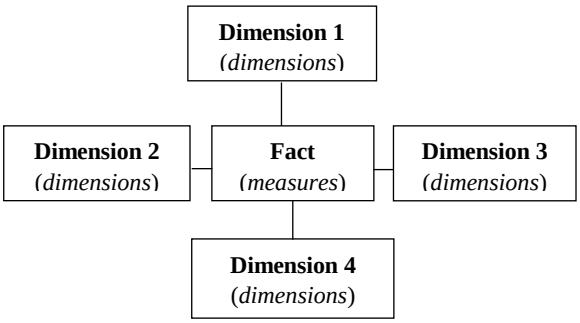

Fig. 3: Star model [22]

B. Snowflake Schema

A snowflake is defined as a relational database model that is utilized for containing measures, dimensions and sub-dimensions in a data mart by applying normalization on dimension tables. A Fact table is surrounded by a number of dimension tables that are directly linked to sub-dimensions (hierarchies).

Fig. 4 shows a snowflake with four dimension-tables, one fact table and one sub-dimension table.

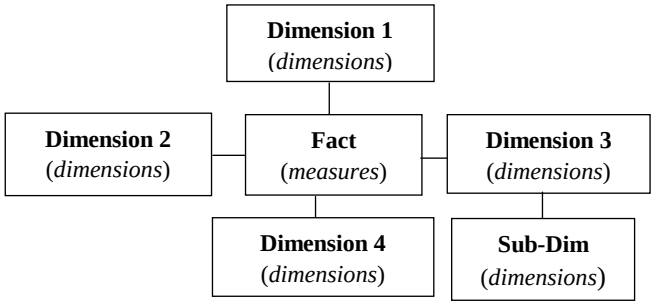

Fig. 4: Snowflake model [22]

The above discussion presents an answer to our **RQ1**.

III. Methodology

In [15] challenges of DW projects failure have been addressed and a framework to facilitate the formalization of requirements elicitation have been suggested. Consequently, our methodology stems from such previous work [15] that combined the bottom-up approach and top-down approach with aim of having one set of requirements that we call here requirements definition, which will be useful for generating the two models. However, our methodology aims at comparing two design models starting with the same requirements (refer Fig. 5).

Concepts from each model are identified and linked to system requirements. A recommendation on a suitable model is made following a comparison.

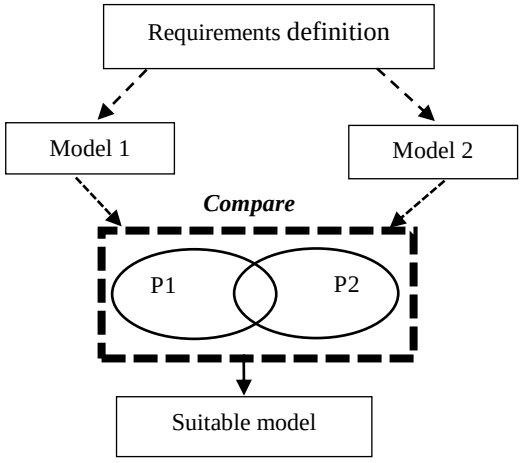

Fig. 5: Methodology [15]

Next, we present the case study used in this paper.

IV. Case Study

Although a DW system as a decision-making support system [24] portrays a dynamic character, we consider, for the purposes of this paper a snapshot of a problem at a certain time to investigate static aspects of the system at the conceptual level.

A. Requirements Definition

Example 1:

Consider the following business requirement and business objective for a data warehouse system.

Business requirement: Model a car-rental company that offers car rental to customers at its agencies.

Business objective: Design a DW system for a decision maker interested in analyzing car-renting facts regarding a certain date in terms of revenue.

Moving to a requirements specification, assume the data warehouse designer has to:

1. Define all entities/objects that would be involved in the design of the above support system.
2. Describe all attributes for each entity/object as well as the relationships among them.

Suppose the designer decides on a semi-formal specification via a UML class diagram containing attributes, relationships and cardinalities to illustrate the static aspect of the above data warehouse system.

A brief presentation of some well-known UML static constructs is presented next.

B. UML Overview

As may be observed in any number of texts, e.g. [25], UML is a graphical language used for the specification, visualization and documentation of software systems. UML diagrams are utilized to model business processes and systems; specifically, the class diagram plays a prominent role in the design phase of (e.g.) mission critical systems [12].

A class diagram is a static representation of a finite set of identified classes (entities) of a system using boxes with three cells containing the name of the class at the top, a list of fields or attributes in the middle and finally a list of operations in the last cell to depict methods (refer Fig. 6). Lines between the boxes in a class diagram represent relationships also called associations. Denotations of relationships among classes are numerous, e.g. association, aggregation, composition, dependency, generalization, etc.

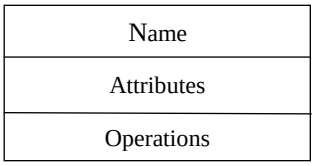

Fig. 6: Class layout [26]

Associations are categorized as common associations, aggregations and compositions. An association end has a role name with a multiplicity constraint attached to classes at both ends [27].

The multiplicity constraint of an association is defined as the number of times that instances of a class may be associated with an instance of another class, represented by a range of non-negative integer values (lower value … upper value) but also represented by the character '*' designating an "unlimited" number of instances. An aggregation is a form of association having a diamond as an indicator at the association end attached to the whole object of the whole-part relationship to describe the whole-part relationship between objects [28]. Since we

concentrate on static aspects of data warehouses in this paper, further discussion of operations and methods (cf. dynamic aspects) is beyond the scope of this paper.

Next, we present the UML class diagrams obtained from the business requirement and objective as per Example 1 for each of a Star and Snowflake models introduced above.

C. Star and Snowflake for Class Diagrams

According to [29] and [30], the fact table is modeled as a composite class having shared-aggregation relationships with the corresponding dimension tables in the diagram and common associations are used between dimension tables, with sub-dimension tables as relationships. The cardinality or multiplicity is the number marked on the links (relationships) of tables (1 to represent one, 1…* to represent one or more and 0…* to represent zero or more).

At the conceptual level of DW systems, surrogate keys (SKs) are used to alleviate latency as DW systems are built for performance improvement. An SK joins the fact and dimension tables, similar to a foreign key being a primary key to one relation (table) and an attribute in another relation. An SK is usually an integer for quick access.

1) Class Diagrams using Star Model

As indicated, the star model consists of a fact table and dimension tables with all the dimension tables directly related to the fact table. This model applies the denormalization principles over all the tables' structure, i.e. none of the tables in a star has been normalized. The star structure for our example appears in Fig. 7.

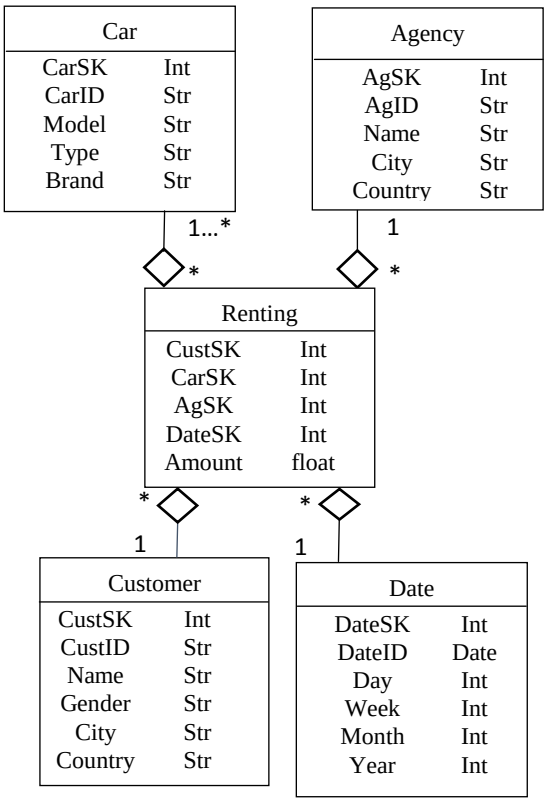

Fig. 7: Model 1 [29]

The *star* representation for ***EXAMPLE 1*** yields five (5) classes – the fact table Renting, and Car, Agency, Customer and Date classes as dimension tables. Renting is modeled as a composite class having shared-aggregation relationships with the corresponding dimension tables [29]; the relationship between the fact table and dimension tables is the aggregation relationship with multiplicities. Referential integrity constraints are maintained via the surrogate keys.

2) Class Diagrams using Star Model

The snowflake model consists of a fact table and dimension tables with sub-dimension tables. This model instead adheres to normalization principles, but only on the dimension tables to obtain sub-dimension tables; the fact table is not affected by the principle.

A *snowflake* representation for Example 1 appears in Fig. 8.

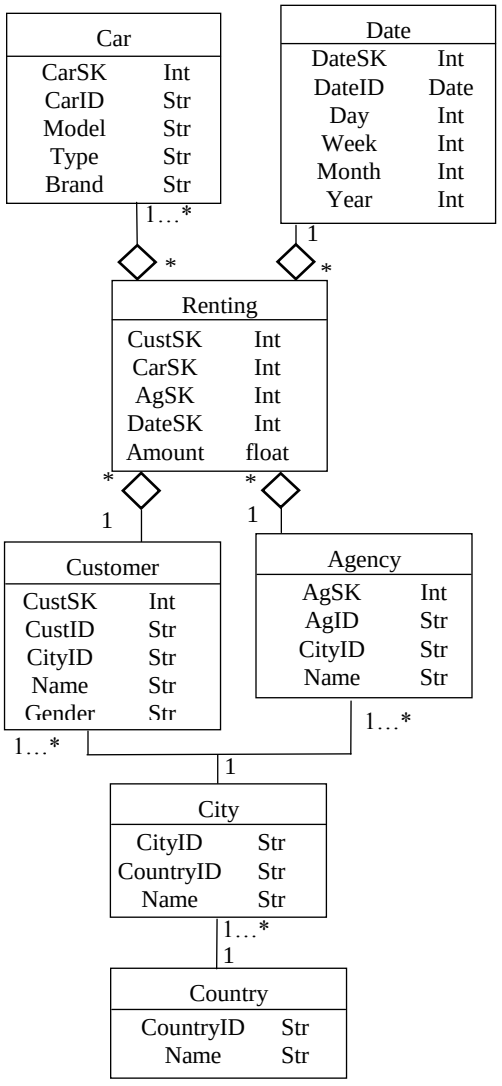

Fig. 8: Model 2 [29]

The snowflake representation in Fig. 8 consists of seven (7) classes: Renting as the fact table and Car, Agency, Customer, City, Date and Country as dimension tables. A similar aggregation as before is defined plus common associations between dimension and sub-dimension tables. The fact table is modeled as a composite class having shared-aggregation relationships with the corresponding dimension tables [29]. The

relationship between dimension tables is the common relationship also known as association [31] and this representation is based on referential integrity as discussed before.

The following section presents a discussion on the comparison of the two models – star and snowflake –using items based on their semantical features.

D. Evaluation and Comparison

Model comparison is an endeavor, which asks for designating semantic correlations between items of the two models [16] [17].

Qualitative model comparison is time consuming and error prone owing to the differences in design decisions [17]. Comparison of the two models in the car rental case study (Example 1) is performed on the strength of the items and features in TABLE , assuming that the items needed for the modeling be generated in the light of stakeholders and decision maker's expectations and needs and adhering to software quality principles. For instance, the requirements defined in the case study state that the proposed system should determine all the necessary elements for representing the system. Therefore, items such as classes; attributes of the class; methods or operations (which are beyond the scope of this paper); and relations between classes may be relevant to describe these requirements based on their semantical features.

We utilize the comparison algorithm in [16] for the evaluation and comparison of our two UML class diagrams. Items possibly relevant for detecting any contradictions, missing or duplicates in the entire system are identified as follows:

Algorithm 1

Input: *A DW Star schema and Snowflake schema;*

Step 1**:** *Pair items from both schemas according to their semantics. It usually involves human contribution;*

Step 2**:** *Calculate the distance between the items of each pair;*

Step 3**:** *Summarize the distances as a model-difference vector to determine the final difference.*

End.

Finally, a vector comprising the distances between relevant item pairs is constructed and its length is evaluated as the resultant difference. As per the list of items identified for the evaluation and comparison, we have three (3) tables containing respective semantical features for each item. The above process is illustrated next.

TABLE I: CLASSES AND INTERFACES DISTANCES [16]

Criteria	Distance
When both models that are semantically equivalent have items with identical names.	0
When both models are semantically equivalent but have items with different names.	0.5
When any one of the models does not have a semantically equivalent class in the other model.	1

We assume a value representing each criterium in TABLE as follows: Zero (0) specifies both classes are identical having the same name; 0.5 indicates both classes have the same name but have some items with different names; and finally 1 indicates that a class of one model has a different name and items from another class of the other model. The values assigned to distances of each criterium is executed in ***Step 1*** of **Algorithm 1**.

TABLE II: ATTRIBUTES OF THE CLASS FEATURES [16]

Feature	Criteria	Value
a	Difference between access modifiers of relevant attributes of the class	Identical: 0 Different: 1
s	Static modifier flag	Identical: 0 Different: 1
n	Difference between names	Identical: 0 Different: 1
t	Difference between attribute type	Identical: 0 Different: 1

The features in TABLE II are as follows: **a** indicates attributes (of the class), **s** indicates a static (modifier), **n** stands for name and **t** indicates type (of the attribute).

A vector is assigned the distances between the class attributes features and its length (*Len*) is determined, by formula (1) below.

$$Len < a|s|n|t > \quad (1)$$

Formula (1) is executed in ***Step 2*** of **Algorithm 1**.

TABLE III: RELATIONS FEATURES [16]

Feature	Criteria	Value
s	Relation source – whether relation into the semantically equal class is outgoing in both models	Identical: 0 Different: 1
t	Relation target – whether relation into the semantically equal class is incoming in both models	Identical: 0 Different: 1
y	Difference between relation	Identical: 0 Different: 1
m	Difference between multiplicity	Identical: 0 Different: 1

Distances between relation features are calculated as a 2nd vector with different parameters by applying ***Step 2*** of the algorithm. Its length is evaluated as in formula (2) below.

$$Len < s|t|y|m > \quad (2)$$

Having compared the identified item pairs, the set of distances between them (to become a set of values later on) is converted into an *n*-dimensional model difference vector where *n* represents the identified item pairs' number. However, the final model difference estimation is a scalar representing the length of the model difference vector (formula 3). The calculation is performed in ***Step 3*** of the algorithm.

$$l = \sqrt{\sum_{i=1}^{n} x_i^2} \quad (3)$$

where x_i represents the distance between item pairs.
Should any of the attributes of a class diagram be missing, or an entire class diagram is missing, all the features of the attributes of the class will be set to 1. Also, access- and static modifiers are omitted and the distance between them is set to 0.

Formula 4 calculates the length for each item pair using the model difference of both schemas.

$$\sqrt{\sum_{i=1}^{46} x_i^2} = \sqrt{(192 + 2)} = 13.93 \approx 14 \qquad (4)$$

The results are shown in TABLE IV

TABLE IV: ITEMS PAIRS COMPARISON FOR MODEL 1 AND MODEL 2 [16]

Diagram 1 Items	Diagram 2 Items	Distance
Car	Car	0
Car.CarSK	Car.CarSK	Len(⟨0|0|0|0⟩) = 0
Car.CarID	Car.CarID	*Len* (⟨0|0|0|0⟩) = 0
Car.Model	Car.Model	*Len* (⟨0|0|0|0⟩) = 0
Car.Type	Car.Type	*Len* (⟨0|0|0|0⟩) = 0
Car.Brand	Car.Brand	*Len* (⟨0|0|0|0⟩) = 0
Date	Date	0
Date.DateSK	Date.DateSK	*Len* (⟨0|0|0|0⟩) = 0
Date.DateID	Date.DateID	*Len* (⟨0|0|0|0⟩) = 0
Date.Day	Date.Day	*Len* (⟨0|0|0|0⟩) = 0
Date.Week	Date.Week	*Len* (⟨0|0|0|0⟩) = 0
Date.Month	Date.Month	*Len* (⟨0|0|0|0⟩) = 0
Date.Year	Date.Year	*Len* (⟨0|0|0|0⟩) = 0
Renting	Renting	0
Renting.CustSK	Renting.CustSK	*Len* (⟨0|0|0|0⟩) = 0
Renting.CarSK	Renting.CarSK	*Len* (⟨0|0|0|0⟩) = 0
Renting.AgSK	Renting.AgSK	*Len* (⟨0|0|0|0⟩) = 0
Renting.DateSK	Renting.DateSK	*Len* (⟨0|0|0|0⟩) = 0
Renting.Amount	Renting.Amount	*Len* (⟨0|0|0|0⟩) = 0
Agency	Agency	0
Agency.AgSK	Agency.AgSK	*Len* (⟨0|0|0|0⟩) = 0
Agency.AgID	Agency.AgID	*Len* (⟨0|0|0|0⟩) = 0
Agency.Name	Agency.Name	*Len* (⟨0|0|0|0⟩) = 0
Agency.City	-	*Len* (⟨1|1|1|1⟩) = 4
Agency.Country	-	*Len* (⟨1|1|1|1⟩) = 4
Customer	Customer	0
Customer.CustSK	Customer.CustSK	*Len* (⟨0|0|0|0⟩) = 0
Customer.CustID	Customer.CustID	*Len* (⟨0|0|0|0⟩) = 0
Customer.Name	Customer.Name	*Len* (⟨0|0|0|0⟩) = 0
Customer.Gender	Customer.Gender	*Len* (⟨0|0|0|0⟩) = 0
Customer.City	-	*Len* (⟨1|1|1|1⟩) = 4
Customer.Country	-	*Len* (⟨1|1|1|1⟩) = 4
-	City	1
-	City.CityID	*Len* (⟨1|1|1|1⟩) = 4
-	City.CountryID	*Len* (⟨1|1|1|1⟩) = 4
-	City.Name	*Len* (⟨1|1|1|1⟩) = 4
-	Country	1
-	Country.CountryID	*Len* (⟨1|1|1|1⟩) = 4
-	Country.Name	*Len* (⟨1|1|1|1⟩) = 4
Aggregation (Date → Renting)	Aggregation (Date → Renting)	*Len* (⟨0|0|0|0⟩) = 0
Aggregation (Car → Renting)	Aggregation (Car → Renting)	*Len* (⟨0|0|0|0⟩) = 0
Aggregation (Agency → Renting)	Aggregation (Agency → Renting)	*Len* (⟨0|0|0|0⟩) = 0
Aggregation (Customer → Renting)	Aggregation (Customer → Renting)	*Len* (⟨0|0|0|0⟩) = 0
-	Association (City ↔ Country)	*Len* (⟨1|1|1|1⟩) = 4
-	Association (City ↔ Agency)	*Len* (⟨1|1|1|1⟩) = 4
-	Association (City ↔ Customer)	*Len* (⟨1|1|1|1⟩) = 4

V. FINDINGS AND RECOMMENDATION

The rounded-up integer value from formula (4) indicates that model 1 is different from model 2. From TABLE IV and the procedure in [16], we infer that the more the value obtained is larger; the more the difference between models is also greater.

TABLE IV indicates that model 1 (star) has no contradictions or duplications of items over model 2 (snowflake), but it has some missing items with respect to model 2 in the sense of classes; attributes of the class; and relations features. Conversely, model 2 only has some items missing with respect to model 1 in the sense of a few attributes of the class features. This makes model 1 the preferred one for a DW system design, owing mainly to the easy-to-read aspects because of fewer items.

The above discussion answers our ***RQ2***.

VI. CONCLUSION AND FUTURE WORK

This paper illustrated through a data mart case study, a methodology to evaluate and compare two DW schemas constructed from the same set of requirements expressed by stakeholders and decision makers. The methodology proposes identifying a list of items needed from a satisfactory model and relating them to the system requirements. These items were compared to determine the more suitable model. To this end, the star- and snowflake models of DW design were considered. Our case illustrated that the star model was more suitable than the snowflake model for the design of data warehouse systems. The star schema resulted in fewer components, leading to reduced complexity. These are important findings especially when we consider the manual generation of DW schemas by a human designer. Such techniques are applied usually during the first phases of a software development methodology. Reduced complexity is furthermore an important consideration in query optimization for very large data warehouses.

As future work, we intend to investigate formal specification aspects of both models. This could be pursued by extending the methodology in Fig. 5 to allow for the reasoning about properties of the models.

REFERENCES

[1] W. H. Inmon, "Building the Data Warehouse". In Chemistry (4th ed). Wiley, 2005.

[2] S. Mathur, G. Sharma and A. K. Soni, "Requirement Elicitation Techniques for Data Warehouse Review Paper". *International Journal of Emerging Technology and Advanced Engineering*, *2*(11), 456–459,

2012. Retrieved from https://www.researchgate.net/profile/Girish_Sharma4/publication/259827907_IJETAE_1212_84/links/02e7e52e0dc392211f000000.pdf

[3] N. H. Z. Abai, J. H. Yahaya and A. Deraman, "User Requirement Analysis in Data Warehouse Design: A Review". *Procedia Technology*, *11*, 801–806, 2013. https://doi.org/10.1016/j.protcy.2013.12.261

[4] M. El Mohajir and I. Jellouli, "Towards a Framework Incorporating Functional and Non Functional Requirements for Data Warehouse Conceptual Design". *IADIS International Journal on Computer Science and Information Systems*, *9*(1), 43–54, 2014. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.640.5590&rep=rep1&type=pdf

[5] A. Nasiri, E. Zimányi and R. Wrembel, "*Requirements Engineering for Data Warehouses"*. 49–64, 2015. Retrieved from http://code.ulb.ac.be/dbfiles/NasZimWre2015incollection.pdf

[6] F. Di Tria, E. Lefons and F. Tangorra, "GrHyMM: A Graph-Oriented Hybrid Multidimensional Model". *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6999 LNCS*, 86–97, 2011. https://doi.org/10.1007/978-3-642-24574-9_12

[7] M. Thenmozhi and K. Vivekanandan, "Data Warehouse Schema Evolution and Adaptation Framework Using Ontology". *International Journal on Computer Science and Engineering (IJCSE)*, *6*(07), 232–246, 2014.

[8] R. Jindal and T. Shweta, Comparative Study of Data Warehouse Design Approaches : A Survey. *International Journal of Database Management Systems*, *4*(1), 33–45, 2012. https://doi.org/10.5121/ijdms.2012.4104

[9] M. Golfarelli and S. Rizzi, "Designing the Data Warehouse: Key Steps and Crucial Issues". *Computer Science and Information Management*, *2*(3), 88–100, 1999. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.124.1900&rep=rep1&type=pdf

[10] R. M. Bruckner, "Developing Requirements for Data Warehouses Systems With Use Cases". *Seventh Americas Conference on Information Systems*, 329–335, 2001.

[11] A. Adesina-Ojo, "Towards the Formalisation of Object-Oriented Methodologies". University of South Africa, 2011. https://doi.org/10.1145/2072221.2072252

[12] M. Singh, A. K. Sharma and R. Saxena, "An UML + Z Framework for Validating and Verifying the Static Aspect of Safety Critical System". *International Conference on Computational Modeling and Security*, 352–361, 2016. https://doi.org/10.1016/j.procs.2016.05.243

[13] N. A. Zafar, "Formal Specification and Verification of Few Combined Fragments of UML Sequence Diagram". *Springer - Arab J Sci Eng*, *41*, 2975–2986, 2016. https://doi.org/10.1007/s13369-015-1999-9

[14] C. Dongmo and J. A. van der Poll, "Evaluating Software Specifications by Comparison". *SAICSIT '11, October 3-5, 2011, Cape Town, South Africa* https://doi.org/10.1145/2072221.2072232

[15] I. N. Mbala and J. A. van der Poll, "Towards Specification Formalisms for Data Warehousing Requirements Elicitation Techniques". *The 3rd International Conference on Computing Technology and Information Management (ICCTIM 2017)*, (1), 45–58, 2017.

[16] O. Nikiforova, K. Gusarovs, L. Kozacenko, D. Ahilcenoka and D. Ungurs, "An Approach to Compare UML Class Diagrams Based on Semantical Features of Their Elements". *The Tenth International Conference on Software Engineering Advances*, (November 2015), 147–152. https://doi.org/10.13140/RG.2.1.3104.4889

[17] M. A. Al-khiaty and M. Ahmed, "UML Class Diagrams : Similarity Aspects and Matching". *Lecture Notes on Software Engineering*, *4*(1), 41–47, 2016. https://doi.org/10.7763/LNSE.2016.V4.221

[18] J. Trujillo, J. Gomez and I. Song, "*Designing Data Warehouses with OO Conceptual Models"*. pp. 66–75, 2001. University of Alicante https://doi.org/10.1109/2.970579

[19] S. Luján-Mora, J. Trujillo and P. Vassiliadis, "Advantages of UML for Multidimensional Modeling". *ICEIS 2004 - Proceedings of the Sixth International Conference on Enterprise Information Systems*, 298–305, 2004. https://doi.org/10.5220/0002633302980305

[20] S. Luján-mora, "*Data Warehouse Design with UML PhD Thesis"*. University of Alicante, 2005.

[21] B. P. Basaran, "*A Comparison of Data Warehouse Design Models"*. MSc Dissertation, Atilim University, 2005.

[22] T. Oketunji and O. Omodara, "*Design of Data Warehouse and Business Intelligence System"*.(Blekinge Institute of Technology), 2011. Retrieved from http://www.diva-portal.org/smash/record.jsf?pid=diva2:831050

[23] M. Golfarelli, "From User Requirements to Conceptual Design in Data Warehouse Design". *Data Warehousing Design and Advanced Engineering*, 15, 2010. https://doi.org/10.4018/978-1-60566-756-0.ch001

[24] S. Haag and M. Cummings, "*Management information systems for the Information Age"*. 9th edition. McGraw-Hill Irwin, 2013.

[25] K. Moremedi and J. A. van der Poll, "Towards a Comparative Evaluation of Text-Based Specification Formalisms and Diagrammatic Notations", *International Journal of Data Mining, Modelling and Management (IJDMMM)*, 11(3), 259 – 283, 2019. Inderscience Enterprises Ltd. https://doi.org/10.1504/IJDMMM.2019.100386

[26] P. Moura, R. Borges and A. Mota, "*Experimenting Formal Methods through UML"*. pp. 1–13, 2015.

[27] S. Kim. and D. Carrington, "Formalizing the UML Class Diagram Using Object-Z". *LNCS 1723, Springer-Verlag Berlin Heidelberg 1999*, 83–98, 1999. https://doi.org/10.1007/3-540-46852-8_7

[28] M. Shroff and R. B. France, "Towards a Formalization of UML Class Structures in Z". *Proceedings Twenty-First Annual International Computer Software and Applications Conference (COMPSAC'97)*, 646–651, 1997. https://doi.org/10.1109/CMPSAC.1997.625087

[29] Y. Mai, J. Li and H. L. Viktor, "Dimensional Modeling for Data Warehouse". *Management Information Systems*, 201–210, 2004.

[30] N. El-Gamal, "Data Warehouse Conceptual Modeling Approaches". *Proceedings of the 37th International Conference On*, (January 2007), 231–242. Retrieved from http://www.researchgate.net/publication/230854949_DATA_WAREHOUSE_CONCEPTUAL_MODELING_APPROACHES/file/79e415056f8261c932.pdf

[31] J. N. Mazón and J. Trujillo, "A Hybrid Model Driven Development Framework for the Multidimensional Modeling of Data Warehouses". *ACM SIGMOD Record*, *38*(2), 12–17, 2009. https://doi.org/10.1145/1815918.1815920

Isaac N. Mbala is an MSc Candidate in Computer Science with School of Computing, University of South Africa (UNISA), South Africa. He obtained his BSc Hons (Computer Science) from the University of Kinshasa (UNIKIN). He has worked for a Vodacom Congo company as Management Information System (MIS) Specialist from 2013 to 2015 in the DRC. He was in charge of Business Intelligence, Development and Reporting. He also worked for JB Capital Partners company as IT Specialist – Software Engineer from 2019 to 2020. He was in charge of Coding, Testing and deployment of the software applications.

His current focus area is to determine the extent to which the use of formal methods for data warehouse systems may alleviate failures within the design process. He is looking forward to do his PhD in Computer science.

Prof John A. van der Poll obtained a BSc from the University of Stellenbosch, South Africa in 1980 and a Hons BSc (Computer Science) in 1982, also from Stellenbosch. He obtained an MSc (Computer Science) from Unisa (University of South Africa) in 1987 and a PhD in Computer Science from Unisa in 2001. He was employed in the Unisa School of Computing from 1988 to 2013. He became a full Professor in Computing in 2007 and moved to Unisa's Graduate School of Business Leadership (SBL) in Midrand in June 2013.

His research interests are in the construction of highly dependable software for Business ICTs, specifically the formal specification and subsequent reasoning of the properties of Business Intelligence applications, Data warehousing, the IoT, IT Governance, and aspects of the 4th Industrial Revolution. He is a Research Professor at the SBL, an NRF rated researcher, category C2 and supervised numerous Master's and Doctoral students to completion.