Evaluating the Reusability of OLAPSec Framework: A Multilayer Security Framework for OLAP Systems

Ahmad Mousa Altamimi
Computer Science, Applied Science Private University
Amman, Jordan

AND

Mahmood Ghaleb Al-Bashayreh
Computer Science, Applied Science Private University
Amman, Jordan

ABSTRACT

Securing the access to Online Analytical Processing (OLAP) systems have become an increasingly important as such systems almost always house confidential and sensitive data that must be restricted to authorized users. A primary limitation of current security systems is that they are designed on top of a very specific data model. This can produce complex and error prone mappings to the elements of other OLAP systems. In addition, constructing comprehensive security systems that allow defining and associating security with an intuitive conceptual OLAP data model is a very complex process. To simplify the development process of security systems, one can consider the using of application frameworks. In this regards, many studies have been proposed frameworks to achieve the aforementioned goal. However, none of them has considered the reusability in its design or development process despite the fact that reusability is one of the main characteristics of successful application framework designs. Having said that, OLAPSec framework was developed to simplify the development process taking into account the reusability concept. This paper investigates the design of this framework in terms of the reusability evaluation method. To this end, empirical experiments have been conducted using well-known tools. Results showed that the framework design satisfies the reusability concept.

Keywords: OLAP, OLAPSec Framework, Reusability Evaluation, Security Framework.

1. INTRODUCTION

Security is an essential demand for all database domains including Data Warehouse (DW) and Online Analytical Processing (OLAP). Organizations gather data and make promises to keep the data secret. Nevertheless, keeping one's promises is usually easier said than done. According to a recent survey of data breaches conducted by the Identity Theft Resource Center [1], an increasing number of incidents is recorded in recent years. This including credit card, medical records, or social security numbers that have been lost to criminals or exposed issues.

In fact, DW and OLAP systems are the most important targeted victims in this context as they are used to gathering data from various sources for analyzing and reporting. The collected data, however, usually contains sensitive or classified data that must be closed for authorized users only. Accordingly, various strict security systems have been created for ensuring secured accessing, for example [2]-[5]. Regardless of the advantages of utilizing such security systems, their developing process is very complex because of two reasons: firstly, OLAP system relies on a different database schema (Data Cube) that involves the aggregation of mixed data. Unauthorized users can issue queries with aggregation functions, such as SUM, AVERAGE, MIN, MAX, then utilize the results to expose the sensitive data. Secondly, OLAP systems can be implemented using different DBMS environments. For example, Row-store (e.g., PostgreSQL) and Column-store (e.g., MonetDB) are two different database management systems that are widely used in decision making. These systems need compatible security systems that can be adapted with little modifications. Hence, in order to reduce the complexity of developing security systems, one can consider the using of software reuse technique for simplifying the development process [6]-[8].

Software reuse is the process of building new software from the use of existing software rather than from scratch. This can be done by building a reusable framework [6], [7] that reduces cost [9] and development time [10]. Consequently, the developer productivity is increased [11], lines of code [10] are decreased, and maintenance efforts [12] are reduced. To the best of our knowledge, no framework has been developed in the literature to be considered as state-of-art for developing OLAP security system. Therefore, the purpose of this paper is threefold. To study the work that relates to software reuse and application frameworks. To examine the design of OLAPSec framework considering the reusability concept. Finally, to evaluate the presented framework reusability based on two steps: the design guidelines step and the software metrics implementation step.

The rest of this paper is organized as follows: the related work is discussed in section 2. The proposed methodology of this research is introduced in Section 3. The design of OLAPSec framework is presented in Section 4 followed by the reusability evaluation in Section 5. Finally, the conclusion and future works are presented in Section 6.

2. RELATED WORK

Researchers have been attempted to maximize the reuse of software rather than to expend the cost of creating something new long time back. Lanergan and Poynton at Raytheon and of Matsumoto at Toshiba [13] were the first who coined the reuse concept. The term "software reuse" refers to the using of some software in more than one software project to improve the developer productivity and the overall quality of software and reducing the costs [14].

To accomplish this goal, authors classified the reusing software into many areas including: reusing data structure [15], [16]; reusing of architecture [17]; reusing design [18]; reusing programs and common systems [19]; reusing modules [20]. However, the ideal reuse technique for our research is the application framework for many reasons: according to [9], [21], [22], the application framework captures the core of reuse techniques in order to achieve maximum [23] large-scale reuse [9], [22], [24]. For instance, it allows the reuse of software design [24]-[26], which includes the architectural and the non-architectural designs [27]. On the one hand, the architectural design or high-level design covers all visible design decisions made by architects to meet the quality attributes and behavioral requirements of the system. On the other hand, the non-architectural design (detailed design) covers invisible design decisions made by developers, such as the selection of a specific algorithm [27].

In the context of privacy and security, providing security countermeasures becoming a strong concern for most organizations. Several models have been defined such as: data encryption [28],[29], which is considered as the main technique for achieving security. However, implementing such technique requires very high cost, complexity, and causes performance degradation [30]. Others models focused on the design process itself, the Unified Modeling Language for example has been extended to define the security constraints [31].

Other researches define the constraints at an early stage of design process [32], and for the entire Data Warehouse life cycle [33]-[35]. Their focus were on the design methodologies that would most effectively use existing technologies, such as, Model Driven Architecture (MDA) and the standard Software Process Engineering Metamodel Specification (SPEM). That being said, application frameworks can provide an optimized solution to design reusable security systems as it addresses business activities in related applications in specific domains [22]. In other words, the primary concepts related to identifying security countermeasures for OLAP domain can be abstracted using collections of interfaces and concrete classes that can be reused and extended each time a new OLAP system is required to be developed with the required security countermeasures as a part of the designing process of decision support systems.

3. METHODOLOGY

Our primary objective in this article is to evaluate the reusability of the OLAPSec framework. To accomplish this, two steps were followed: (1) framework design guidelines application; (2) software metrics implementation. First, framework design guidelines (rules) were applied as an approved method to confirm frameworks reusability [36]. Accordingly, the FxCop, a static code analysis tool, was used to analyze the compiled code based on seven well-known guidelines [36]. A brief description of these guidelines is given in Table 1. Using FxCop, 211 possible code violations of the guidelines were checked in an iterative process. As a result, a number of problems were discovered and solved to optimize the OLAPSec design and implementation.

TABLE I
DESCRIPTION OF THE FXCOP DESIGN GUIDELINES

Guideline	Usage
Naming	Used for naming assemblies, namespaces, types, and members in classes
Type	Used for using static and abstract classes, interfaces, enumerations, and structures
Member	Used for designing and using properties, methods, constructors, fields, events, operators, and parameters
Extensibility	Used for extensibility mechanisms such as sub-classing, using events, virtual members, and callbacks
Exceptions	Used for designing, throwing, and catching exceptions
Usage	Used for using common types such as arrays, attributes, and collections, supporting serialization, and overloading equality operators
Common design patterns	Used for choosing and implementing dependency properties and the dispose pattern

Second, software metrics were implemented as an approved quantitative measurement method to confirm frameworks reusability [37], [38]. Accordingly, the well-known Goal Question Metric (GQM) mechanism [39] was used to identify suitable design metrics that provide informative quantitative measurement about the quality of the framework design and implementation [40]. In fact, it has been proven that high maintainability and low coupling (low complexity) have a positive influence on reusability [40]-[43]. Therefore, two metrics were used to measure OLAPSec maintainability and complexity, which are: (1) the maintainability index which is measured based on the OLAPSec classes and interfaces; and (2) the class coupling which is measured based on the OLAPSec classes and interfaces.

4. OLAPSEC FRAMEWORK DESIGN

Fig. 1 shows the proposed architectural design of the OLAPSec framework. The architecture consists of three primary layers: the decision support system layer; the framework layer; and the database engine layer. Our focus here is on the internal architecture of the middle layer (framework layer). This layer follows the well-known "segmented layers" notation [27]. It consists of two sub-layers: the Access Control Layer (ACL) and Inference Control Layer (ICL), with arrows representing the "allowed-to-use" relations between layers and among components within each layer [27].

In the OLAPSec framework, the process is initiated by submitting a query using a standard client query language such as MDX or SQL. After receiving the query, it is translated into an Intermediate Representation (IR) [2] by the Query Translator module. The Access Control Module then receives the query in the IR format and decides whether the query is legitimate to answer by checking the access control rules and policies stored in the security database. Ultimately, the query is assessed for possible inference attacks. This is done by utilizing a specific set of data structures and algorithms. Assuming that the query is acceptable at this stage, it will be converted back to its original format by the IR Converter module for execution.

The execution is carried out by the backend server and results are returned back to the decision support systems on the upper layer. Further information about the OLAPSec framework can be found in [4].It is important to mention here that the framework was designed by applying the SOLID design principles, which were proven to support reusability [44], [45].

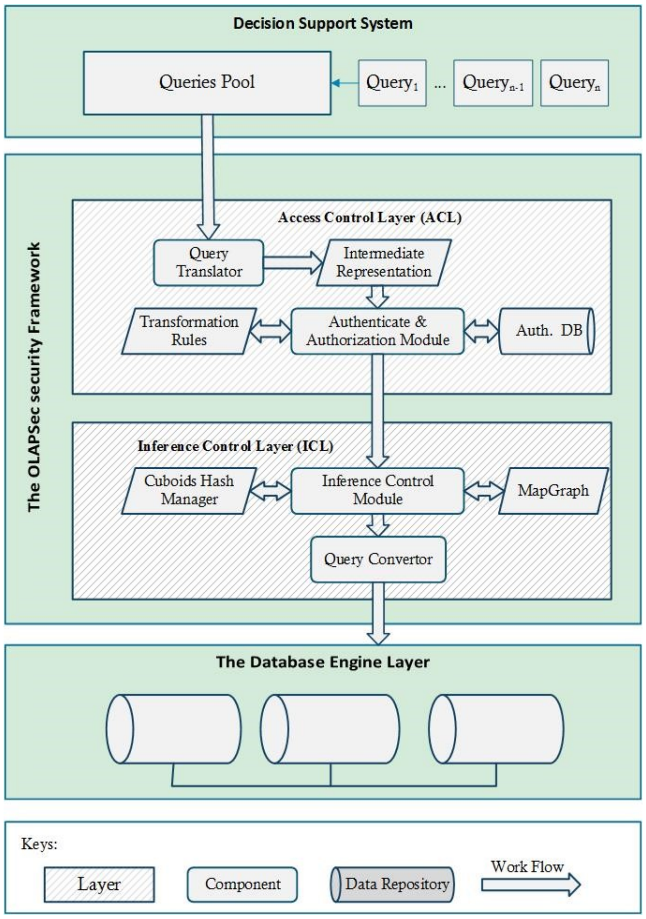

FIG. 1 THE ARCHITECTURAL DESIGN OF THE OLAPSEC FRAMEWORK

5. OLAPSEC FRAMEWORK EVALUATION

A presentation of the two steps implementation of the OLAPSec evaluation processes, which are the OLAPSec design guidelines step and the software metrics implementation step. In the first step, the FxCop tool was used to analyze the OLAPSec compiled code based on: 61 design rules, 10 globalization rules, 16 interoperability rules, 2 mobility rules, 23 naming rules, 16 performance rules, 3 portability rules, 21 security rules, 20 security transparency rules, and 39 usage rules [36]. The results show that OLAPSec satisfies 94% of these guidelines, which confirms its reusability.

In the second step, the maintainability index and the class coupling matrices were calculated using the Microsoft Code Metrics PowerTool. The maintainability index of the OLAPSec framework was calculated during the design and implementation phases. According to Microsoft [46], the calculated value of maintainability index is classified into three categories, which are: (1) high, between 20 and 100; (2) moderate, between 10 and 19; and (3) low, between 0 and 9. As shown in Table 2, 100% of the calculated values of maintainability index for the OLAPSec framework are classified with high maintainability.

TABLE 2
OLAPSEC MAINTAINABILITY INDEX

Maintainability Index	Classification	Frequency	Percentage
$20 \leq$ value ≤ 100	High	18	100%
$10 \leq$ value ≤ 19	Moderate	0	0%
$0 \leq$ value ≤ 9	Low	0	0%
Total		18	100%

Similarly, the class coupling of the OLAPSec was also calculated during the design and implementation phases. According to Microsoft Code Metrics PowerTool, the calculated values of class coupling are classified into three categories, which are: (1) low, between 0 and 9; (2) moderate, between 10 and 80; and (3) high, greater than 80. As shown in Table 3, 83.3% (15 out of 18 values) of the calculated class coupling of the OLAPSec are classified with low coupling and only 16.7% (3 out of 18 values) of the calculated class coupling are classified with moderate coupling. However, the mean value of the moderate coupling is 17, which is nearer to the minimum threshold of the moderate class coupling value. This represents accepted class coupling and therefore there is no need for design and implementation refactoring.

TABLE 3
OLAPSEC CLASS COUPLING

Class Coupling	Classification	Frequency	Percentage
$0 \leq$ value ≤ 9	Low	15	83.3%
$10 \leq$ value ≤ 80	Moderate	3	16.7%
value ≥ 81	High	0	0%
Total		18	100%

With reference to the calculated metrics, it can be concluded that OLAPSec framework design is considered as reusable.

6. CONCLUSION AND FUTURE WORK

In this paper, the reusability evaluation of OLAPSec framework was presented starting with the framework design guidelines application using the FxCop tool, followed by the software metrics implementation using the Microsoft Code Metrics PowerTool. While the result of the first step demonstrates that OLAPSec satisfies 94% of framework design guidelines, the second step shows that 100% of the calculated values of maintainability index are classified as highly maintainable, and 83.3% of the calculated values of class coupling are classified as low coupling. These measures gave clear insight that the presented framework is reusable and there is no need for design refactoring. However, future research could evaluate the framework reusability by developing different prototypes for decision support systems based on various database management systems environments.

ACKNOWLEDGMENT

The authors are grateful to the Applied Science Private University, Amman-Jordan, for the full financial support granted to cover the publication fee of this research article

REFERENCES

[1] Identity Theft Resource Center survey. Available at: http://www.idtheftcenter.org/itrc-surveys-and-studies.html. Accessed on 23/03/2017.

[2] A, Altamimi, "Securing OLAP Cubes". Lambert Academic Publishing, Germany. ISBN 978-3-659-39290-0. ch 3.

[3] A. Altamimi, "OSMO: An Optimizing and Securing Model for OLAP Queries". International Journal of Academic Research, vol 7, No 4. 2015.

[4] A, Altamimi, T. Eavis, "OLAPSec: A Comprehensive End-to-End Security Framework for OLAP Cubes". International Journal of Information Security, vol 3. 2015.

[5] A, Altamimi, M AlBashayreh, "An Integrable Framework for Securing Multidimensional Data with MDX". The 6th European Conference of Computer Science (ECCS '15), Rome, Italy. 2015.

[6] [6] M. Al-Bashayreh, "A Reusable Application Framework for Context-Aware Mobile Patient Monitoring Systems", Ph.D., Universiti Utara Malaysia, 2014.

[7] W. Zhang and M. Kim, "What Works and What Does Not: An Analysis of Application Frameworks Technology," Journal of Business Systems, Governance and Ethics, vol. 1, pp. 15-26, November 2006.

[8] J. v. Gurp and J. Bosch, "Role-Based Component Engineering," in Building Reliable Component-Based Software Systems, I. Crnkovic and M. Larsson, Eds., ed Boston, MA: Artech House, 2002, pp. 135-154.

[9] D. Parsons, A. Rashid, A. Telea, and A. Speck, "An Architectural Pattern for Designing Component-Based Application Frameworks," Software: Practice and Experience, vol. 36, pp. 157–190, February 2006.

[10] R. Neumann, S. Günther, and N. Zenker, "Reengineering Deprecated Component Frameworks: A Case Study of the Microsoft Foundation Classes," in 9th International Conference on Business Informatics, Vienna, Austria, 2009, pp. 737-748.

[11] U. Kulesza, V. Alves, A. Garcia, C. J. P. d. Lucena, and P. Borba, "Improving Extensibility of Object-Oriented Frameworks with Aspect-Oriented Programming," in Reuse of Off-the-Shelf Components. vol. 4039, M. Morisio, Ed., ed Berlin, Germany: Springer, 2006, pp. 231-245.

[12] D. C. Schmidt, A. Gokhale, and B. Natarajan, "Leveraging Application Frameworks," Queue, vol. 2, pp. 66-75, July/August 2004.

[13] Matsumoto, Y., and Y. Ohno, Japanese Perspectives in Software Engineering, Prentice-Hall, Enlgewood Cliffs, New Jersey, 1989.

[14] Champman, M; Van der Merwe, Alta, "Contemplating Systematic Software Reuse in a Small Project-centric Company", Saicsit 2008, South Africa.

[15] Guoqing Xu, "Finding reusable data structures", Proceedings of the 27th Annual {ACM} {SIGPLAN} Conference on Object-Oriented Programming, Systems, Languages, and Applications, {OOPSLA} 2012, part of {SPLASH} 2012, Tucson, AZ, USA, October 21-25, 2012.

[16] Pradel, Michael, Markus Huggler, and Thomas R. Gross. "Performance regression testing of concurrent classes." Proceedings of the 2014 International Symposium on Software Testing and Analysis. ACM, 2014.

[17] Turk, Dan, Robert France, and Bernhard Rumpe. "Limitations of agile software processes." arXiv preprint arXiv: 1409.6600 (2014).

[18] Lohrmann, Matthias, and Manfred Reichert. "Effective application of process improvement patterns to business processes." Software & Systems Modeling (2015): 1-23.

[19] Solanki, Samar Upadhyay Mr Bharat. "Study of reuse method and techniques in software engineering to reduce and optimize the cost of software." GLOBAL JOURNAL OF MULTIDISCIPLINARY STUDIES 3.8 (2014).

[20] Moon, Suk Hwan, and Cheol sick Lee. "Designing and Embodiment of Software that Creates Middle Ware for Resource Management in Embedded System."International Journal of Software Engineering & Its Applications 8.6 (2014).

[21] R. J. McManus, E. P. Bray, J. Mant, R. Holder, S. Greenfield, S. Bryan, et al., "Protocol for a Randomised Controlled Trial of Telemonitoring and Self-Management in the Control of Hypertension: Telemonitoring and Self-Management in Hypertension," BMC Cardiovascular Disorders, vol. 9, pp. 1-21, February 2009.

[22] N. F. Ahmad, D. B. Hoang, and M. H. Phung, "Robust Preprocessing for Health Care Monitoring Framework," in 11th IEEE International Conference on e-Health Networking, Applications and Services, Sydney, Australia, 2009, pp. 169-174.

[23] E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Reading, MA: Addison-Wesley, 1995.

[24] D. C. Schmidt and F. Buschmann, "Patterns, Frameworks, and Middleware: Their Synergistic Relationships," in 25th International Conference on Software Engineering, Portland, OR, 2003, pp. 694-704.

[25] G. Polancic, R. V. Horvat, and I. Rozman, "Improving Object-Oriented Frameworks by Considering the Characteristics of Constituent Elements," Journal of Information Science and Engineering, vol. 25, pp. 1067-1085, July 2009.

[26] I. Crnkovic, B. Hnich, T. Jonsson, and Z. Kiziltan, "Basic Concepts in CBSE," in Building Reliable Component-Based Software Systems, I. Crnkovic and M. Larsson, Eds., ed Norwood, MA: Artech House, 2002, pp. 3-22.

[27] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, et al., Documenting Software Architectures: Views and Beyond, 2nd ed. Upper Saddle River, NJ: Addison-Wesley, 2011.

[28] S. C. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Over-encryption: Management of Access Control Evolution on Oursourced Data", (VLDB), 2007.

[29] N. Yuhanna, "Your Enterprise Database Security Strategy 2010", Forrester Research, Sep. 2009

[30] Santos, R.J., Bernardino, J. and Vieira, M. "A survey on data security in data warehousing: Issues, challenges and opportunities," In Proceedings of IEEE International Conference on Computer as a Tool (EUROCON), pp.1-4, 2011.

[31] Eduardo Fern´andez-Medina, Juan Trujillo, Rodolfo Villarroel, and Mario Piattini, "Developing secure data warehouses with a uml extension," Inf. Syst., 32(6):826–856, September 2007.

[32] Trujillo Juan, Soler Emilio, Fernndez-Medina Eduardo, and Piattini Mario, “An engineering process for developing secure data warehouses,” Information and Software Technology, 51(6), pp: 1033 – 1051, 2009.

[33] Krishna Khajaria and Manoj Kumar, “Modeling of security requirements for decision information systems,” SIGSOFT Softw. Eng. Notes, 36(5), pp: 1–4, September 2011.

[34] Kumar Manoj, Gosain Anjana, and Singh Yogesh, “Stakeholders driven requirements engineering approach for data warehouse development,” JIPS, 6(3), pp: 385–402, 2010.

[35] Singh Yogesh, Gosain Anjana, and Kumar Manoj, “From early requirements to late requirements modeling for a data warehouse,” Networked Computing and Advanced Information Management, International Conference on, pp: 798–804, 2009.

[36] K. Cwalina and B. Abrams, Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .Net Libraries, 2nd ed. Upper Saddle River, NJ: Addison-Wesley, 2009.

[37] M. Lanza and R. Marinescu, Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems. Berlin, Germany: Springer, 2006.

[38] D. Soni, R. Shrivastava, and M. Kumar, "A Framework for Validation of Object-Oriented Design Metrics," International Journal of Computer Science and Information Security, vol. 6, pp. 46-52, December 2009.

[39] V. R.in Basili and H. D. Rombach, "The TAME project: towards improvement-oriented software environments," IEEE Trans. Softw. Eng., vol. 14, pp. 758-773, June 1988.

[40] L. M. Laird and M. C. Brennan, Software Measurement and Estimation: A Practical Approach. Hoboken, NJ: John Wiley & Sons, 2006.

[41] S. Ghosh, S. K. Dubey, and A. Rana, "Comparative study of the factors that affect maintainability," Int. J. Comput. Sci. and Eng., vol. 3, pp. 3763-3769, December 2011.

[42] D. Galin, Software Quality Assurance. Harlow, England: Pearson Educ. Limited, 2004.

[43] T. Mens and T. Tourwe, "A survey of software refactoring," IEEE Trans. Softw. Eng., vol. 30, pp. 126-139, February 2004.

[44] R. C. Martin, Agile Software Development: Principles, Patterns, and Practices. Upper Saddle River, NJ: Prentice Hall, 2003.

[45] M. Al-Bashayreh, "Domain Model Validation of Context-aware Mobile Patient Monitoring Systems", Procedia Computer Science, vol. 62, pp. 539-546, 2015.

[46] Code Metrics Values. Available at: https://msdn.microsoft.com/en-us/library/bb385914(v=vs.%20100%20).aspx. Accessed on 23/03/2017.