Performance analysis on data access patterns in layered Information Systems, an Architectural Perspective

GholamAli Nejad HajAli Irani[1], Zahra Jafari[2]

[1] Faculty of Engineering, University of Bonab
Bonab, 5551761167, Iran

[2] Department of Computer Engineering, Islamic Azad University, Tabriz Branch
Tabriz, Iran

Abstract

Layered architectures are the main architectures that are used in architecture of Information Systems. Data Access Layer (DAL) is a common layer in all layered architecture of Information Systems. One of the most important steps in architectural design of information systems is designing the DAL. More than 20 Data Access Patterns (DAP) have been developed for DALs. So providing a DAL based on existing DAPs is major activity in architectural design. Therefore an architect should be familiar to all existing DAPs and their evaluation parameters.
Performance is one of most important parameter in architecture evaluation. In this paper, different aspects of performance of existing DAPs have been investigated and analyzed. To obtain this aim, firstly, a new classification has been provided for categorizing all DAPs based on architectures difference of them. Secondly, a simulation program as named DALSim has been developed for measuring performance of all DAPs. Finally, some comparisons of DAPs have been provided based on DALSim results.
Based on provided classification for DAPs and performance comparison results of them, architects of information systems can easily compare and evaluate existing DAPs. With fully development of provided results in other aspect such as modifiability, extensibility, security etc., a complete framework for all DAPs can be provided.
Keywords: *Layered Software Architecture, Layered Information Systems, Data Access Patterns.*

1. Introduction

The most important problem in developing large scale software is its architecture. Architectural styles assist the architects in presenting their architecture and make some simplicity in transition of experiments from experts to the students. Layered architecture as one of the architectural styles, widely used in Information Systems [4]. Different types of layered architectures such as 3 layer architecture [1], 5 layer architecture [2] etc., have been provided as yet. Data Access Layer (DAL) as a common layer exists in all types of layered architecture. There are some various methods to implement DALs that named Data Access Patterns (DAP).
More than 20 DAPs for Information System have been provided. Therefore in the content of DAPs, it is necessary to provide an evaluation framework for all existing DAPs. For providing an evaluation framework for all DAPs, all aspect of them such as extensibility, performance, modifiability, security etc, which are named quality attributes should be considered.
To obtain this aim, some steps as named DALFra-Process have been provided as following:

1- To investigate concepts of data access layer and its responsibilities and other related concepts.
2- To collect and classify all existing DAPs for information systems.
3- To collect quality attributes that affect on different aspects of DAPs.
4- To investigate measurements on each quality attribute and evaluate each of them.
5- To provide a general evaluation framework for all DAPs.
6- To provide guidelines to use and evolution of provided framework.

Performance is one of the important quality attributes in software architecture. In this paper, because of the large size of works to reach that mentioned framework, performance analysis of that has been provided.
To obtain this aim, first of all, step 1 and 2 of DALFra-Process have been done completely. Then a new simulation program as named DALSim has been written. So performance analysis on each DAP have been provided using the DALSim and measurement diagram of performance analysis of each other have been presented as results of this paper.

2. DAL, DAP and Persistence

All layered architectures for Information Systems have used three base layers [1]. Firstly, User-Interface (UI) layer is responsible for communications between outside and inside the software system. Secondly, Business Logic Layer (BLL) performs all logics about the steps of software processes and usecases. Thirdly, Data Access Layer is an interface between Database System (DBMS) and internal layers (BLL and UI). Basically none of the other layers can directly access to DBMS and all functionalities on DBMS should be perform by the DAL. Basic functionalities of DBMS are CRUD functions [5]. So each DAL should perform CRUD functionalities. On the other hand, authentication and authorization of BLL classes should perform by the DAL as well.
Finally, the major responsibilities that each DAP should perform [5] are as following: CRUD functionalities, connecting to DBMS, disconnecting from DBMS, managing code based transactions, handling all DAL exception, etc.
Some architectural styles have been developed for DALs which were named Data Access Patterns. In [6,7] more than 20 patterns have been gathered. Software architects can use each of DAP in their architectures.
Persistence so called Persistence Framework (PF) is software package or components that can be used as a DAL in Information Systems architecture. In [8,9] more than 50 PF can be found.
Persistence and DAPs are different. PFs are software packages similar a platform for DAPs. It means that each DAP can be implement with a PF. For example Hibernate [10] is most famous PF for java platforms and each DAP can be implementing with Hibernate.

3. Data Access Patterns Classification

Software architecture is a fundamental organization about components [11]. More than 20 DAPs have been provided for information systems [6,7]. So, based on architectural perspective and after an investigation in to these patterns we reached to a general classification in a way that every pattern is falling in one of these categories.
DAP0: Patterns with no data access layer. In these patterns all of BLL classes perform their functionalities to DBMS by themselves. Advantages of these patterns are performance which may be used in real time systems. An instant of this category is introduced in [6] by the name of "Transaction Script". The model of DAP0 is shown in figure 1.
DAP0Sp: this is DAP0 with difference that BLL classes use Stored Procedures to fetch data and alter them instead of connecting directly to the database. In enterprise information systems where problem domain is very vast and there are complicated use cases, using this methods will make a big problem in development process.

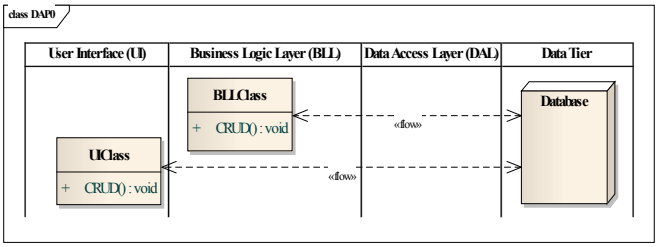

Fig. 1 DAP0, without any DAL.

DAP1: It encapsulates access to the database. In these patterns, using one or more classes in data access layer, implementation details like database name, server name, user name and password are kept hidden from upper layers. Now all of BLL classes use these classes to communicate with database. There are methods defined in DAL classes to perform requests coming from BLL classes including create, read, update and delete (CRUD) operations. This pattern is like "Metadata Mapping" introduced in [6]. In these patterns an interface between BLL and DAL has defined supporting all data access methods and for each database type a class has been created and inherited from that interface [7]. The model of DAP1 is shown in figure 2.
DAP1T: same as DAP1 except that this kind of pattern has two methods (we call them "Begin" and "End") in order to support transaction. These two methods are used to start and finish transactions. CRUD operations are executed after a transaction has started (by calling Begin method) and at last it will be finished (by calling End method). The whole transaction will be done (commit state) or rejected (rollback) if there were errors. In the case of rollback, relevant error message will be presented to user and all changes to database (affiliated by this transaction) will be rolled back.
DAP2: In this pattern, for each table in the project, a class will be created (we call them "Entity" classes) and CRUD operation for that table performed by its class. Advantage is that one can put syntactical and access control on each table separately. This pattern is introduced as "Table data gateway" in [6]. The suggested model of DAP2 is shown in figure 3.
DAP2F1: for setting better control and stronger security and also respecting to modularity principal, this pattern has some improvements to DAP2.
In order to give permission to an entity to select data from other entity classes, this pattern define a "Finder" class for each entity class and gather them in a common layer (accessible to all of entity classes). Finder is responsible

for selecting data from its respective entity and serves all of entity classes [6].
DAP2F2: same as DAP2F1 except that we have just one general Finder class able to select data from all entity tables and serve all entity classes [7].
DAP2F1Sp and DAP2F2Sp: implemented with stored procedures (for higher performance) and DAP2T: supporting transactions.
With the same concept DAP2F1T, DAP2F2T, DAP2F1SpT, DAP2F2SpT can be defined.

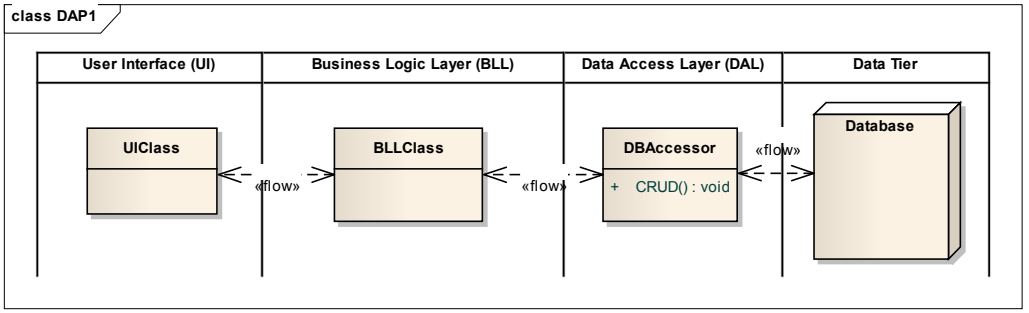

Fig. 2 DAP1: Hiding details of DBMS by a DBAccessor Class.

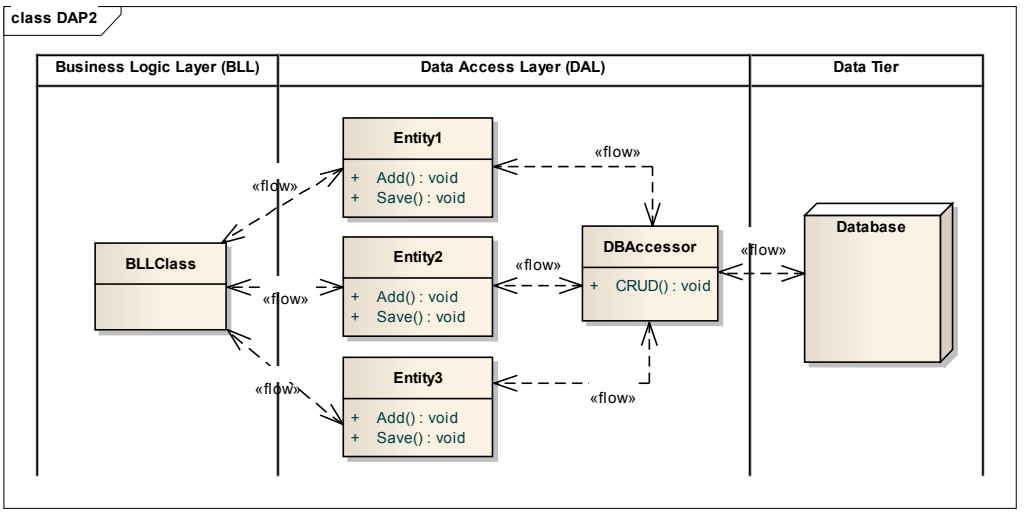

Fig. 3 DAP2, Access to each data via its entity class.

DAP3: similar to DAP2, this pattern is using entity classes for tables. The difference is that respecting to object oriented heuristics all CRUD methods are implemented in a common class and all entity classes inherit these methods from that. Adding support for transactions, implementing with stored procedures and separating Finder classes from entity classes are possible in this pattern to define new patterns. "Object-Relational Metadata Mapper" pattern introduced in [6] falls in this category. The model of DAP3 is shown in figure 4.
DAP3M: same as DAP3 except that all of the Meta-data of the tables and fields are stored and the parent class which all entity classes are inherited from that can check validity of table names, view names, field names and data type and so on, using this Meta-data. Meta-Data holds schema information including the information of tables, views and their ID's. Every module has several tables or views each having names and aliases. Each table or view has several fields. In this pattern parent class is able to create CRUD executable statements dynamically. The model of DAP3M is shown in figure 5.
DAP4M: unlikely DAP2 and DAP3 this pattern doesn't create an entity class for each table. In order to improve extendibility and modifiability this pattern uses Meta-data (like DAP3M) and defines some general classes in DAL to do all tasks. In this pattern when a BLL class wants to do a CRUD operation, sends all information (table name, fields name and values and action) to DAL and then validity of information are checked. If all information were valid, an

executable statement (depending on request) has made dynamically and execute.

"Layer supertype" pattern introduced in [6] is like this pattern. The model of DAP3 is shown in figure 6.

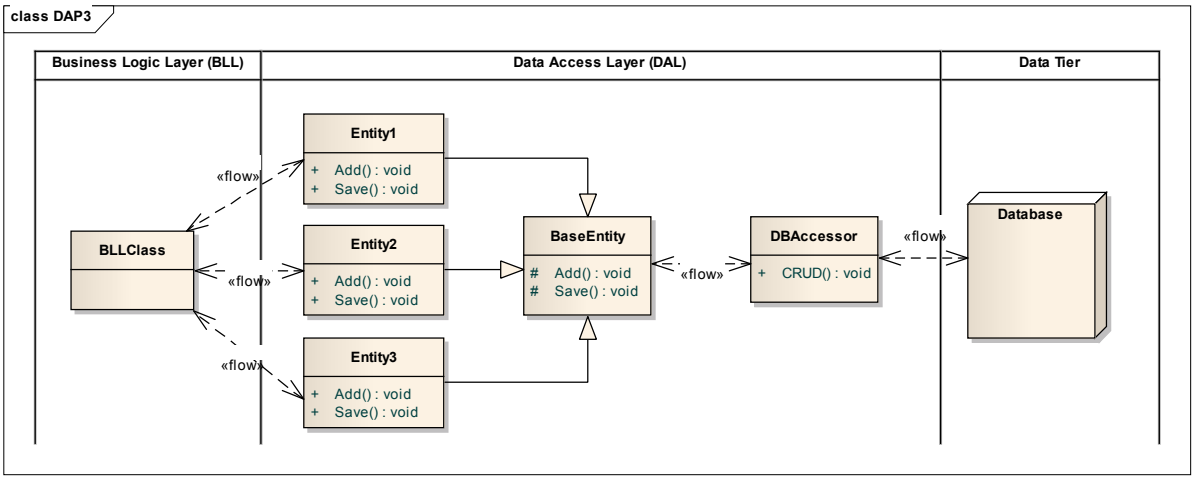

Fig. 4 DAP4: All entity classes inherit from a base entity class.

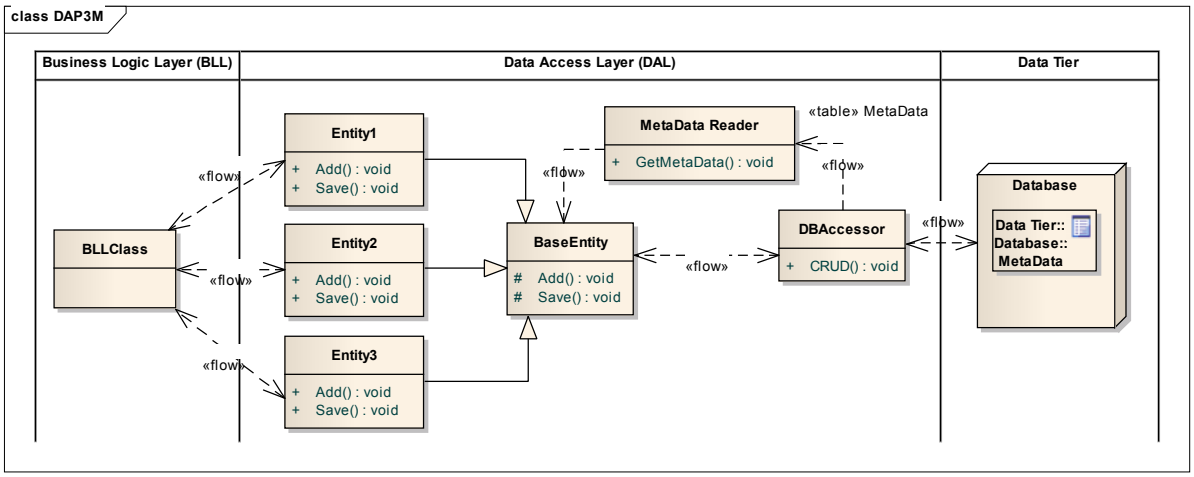

Fig. 5 DAP3M: DAP3 with Meta Data functionality.

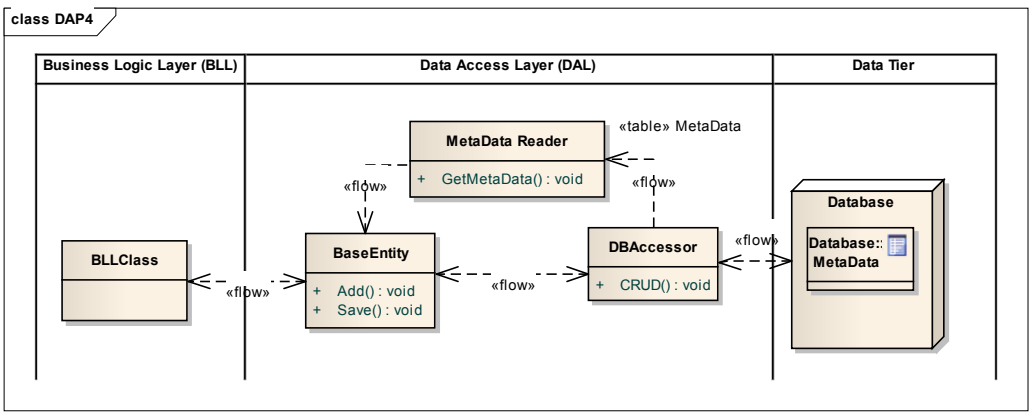

Fig. 6 DAP4, fully meta data based pattern.

All categorized patterns shown in table 1.

Table 1: Provided category for all existing DAPs.

Code	Description
DAP0	These methods don't use any data access layer and any parts of code directly connect to database. Due to high performance, these methods mostly are used in real time systems.
DAP1	These methods use one or more classes in data access layer to encapsulate database access details.
DAP2	These methods use DAP0. Also one entity class is created for each table in database and all CRUD (Create, Retrieve, Update and Delete) methods on this table are handled by its entity class.
DAP3	These methods are similar to DAP2 and entity classes are created for each table, but based on object oriented heuristics, all CRUD methods implemented in one class and other entity classes inherit from it.
DAP3M	These methods like DAP3, but metadata of all tables are stored in database as well. All entity classes inherit from Base Class. It create SQL command dynamically and manage them using stored metadata.
DAP4	In these methods unlike DAP3 and DAP2, entity classes for each table are not created. To gain excellent extendibility and modifiability (like DAP3M), metadata of all tables is stored in database and one or more classes perform all CRUD methods for all tables.

4. Performance Analysis

The time needed to response to an event or which events in one period of time are defined as performance [12]. For measuring performance of DAPs, there are different parameters can be affected. But with the architectural perspective, communications take longer time in comparison with computations. So, just components and relations among them have been considered in measurement of performance.

Two types of components exist in architecture of DAPs. Firstly, existing classed that make the each DAP and secondly components of DBMS that can be Stored Procedures (Sp) or Triggers or Functions that are written in DBMS.

For measuring and comparing performance of existing DAPs, a new simulation application that named DALSim has been developed. All source codes of DALSim are available at [13]. For comparing performance of each DAP, all CRUD functionalities should be implemented. Due to long working of implementing all CRUD functionalities, all DAPs implemented just for Insert operation.

4.1 Architecture Difference

First aspect on analysis of DAPs is difference in their architecture. For analyzing them, some evaluation charts have been provided based on our simulation application (DALSim). In DALSim, the time of running each DAP tested 20 times. So the median number of 20 times has been calculated. In all charts, X-axis is number of Insert operation times and Y-axis the time of Insert operation per second. Figure 7 shows the comparison of one class difference in architecture.

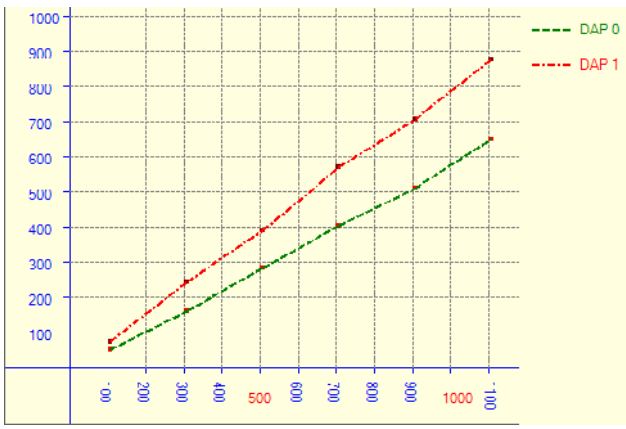

Fig. 7 Evaluation chart for one class difference in architecture.

DAP3 and DAP4 have most classes in comparison with DAP0. MVSDAP [14] as a new pattern that is kind of DAP4 is implemented in DALSim. Figure 8 shows the comparison of architecture in minimum and maximum class difference in architecture.

4.2 Stored Procedure Difference

Second aspect of analysis of DAPs is difference in using the written functions or Stored Procedures in DBMS. By the use of Stored Procedures, Business Login of processes and usecases is transferred into DBMS [3]. In this case, DAL is removed from the architecture and all Stored Procedures in DBMS can directly connect to each data and manipulate them and it is disadvantage of using Stored Procedures.

Because of difference in architecture, new software application as a simulator has been developed and source code of it available at [15]. For evaluating this method, DAP2 as an average mode of classes is compared with

DAP2Sp. Figure 9 shows the evaluation chart of DAP2 and DAP2Sp. In this chart as shown in figure 9, scope of ranges is selected 1000. DAP2Sp is much faster than DAP2, so in 10000 instances, difference is approximately 15 seconds.

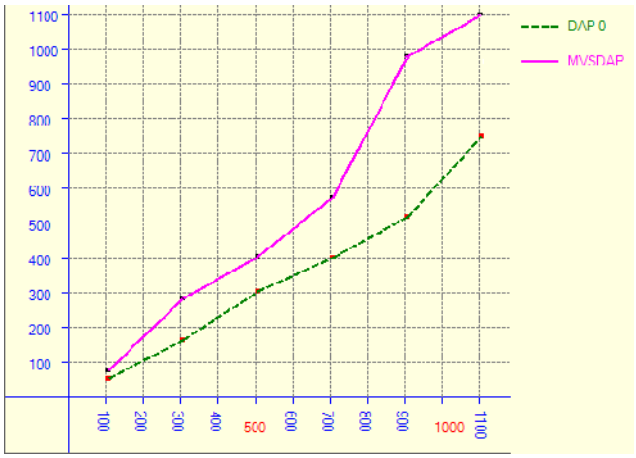

Fig. 8 Evaluation chart for minimum and maximum class difference.

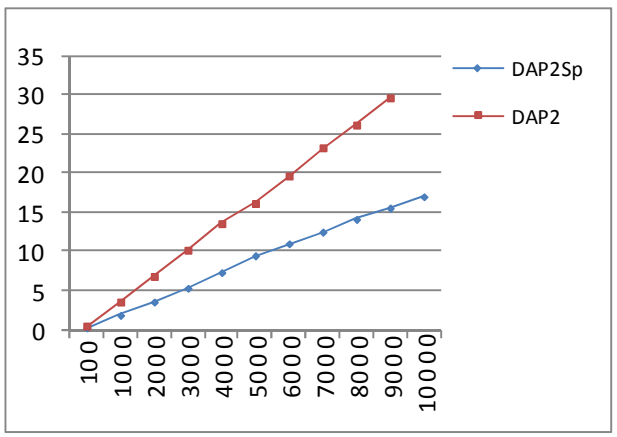

Fig. 9 Evaluation chart for use and non use of Stored Procedures.

5. Conclusion and future works

In this paper towards a general framework for existing DAPs some steps has been suggested. Then in order to step 1 and 2, by the analysis of concepts and the architecture of all DAPs, a new classification for all existing data access patterns has been provided based on architectural perspective and all DAPs can be placed in it.

So, performance as a major quality attribute of any evaluation framework has been selected. Then, for analyzing the performance of DAPs, new simulation software has been developed and performance of existing DAPs has been measured. Afterwards, evaluation charts of performance analysis are shown.

For the future works, by the use of the process used, other quality attributes of DAPs such as extensibility, modifiability, security etc, can be provided. Completing the proposed framework, it can help software architect to evaluate and select their sufficient DAP for their architectures.

References

[1] D. Alur, J. Crupi, D. Malks, Core J2EE Patterns, Best Practices and Design Strategies, Second Edition, Sun Microsystems Inc. 2003.

[2] R. Lhotka, Expert C# 2008 Business Objects, Apress, 2008.

[3] L. Fischer, BPM Excellence in Practice 2009: Innovation, Implementation and Impact Award-winning Case Studies in Workflow and Business Process Management, Future Strategies, Incorporated, 2009.

[4] P. D. Sheriff, Fundamentals of N-Tier Architecture, PDSA, Inc., 2006.

[5] S. K. Rahimi, F. S. Haug, Distributed Database Management Systems: A Practical Approach, IEEE Computer Society, 2010.

[6] M. Fowler, D. Rice, M. Foemmel, E. Hieatt, R. Mee, R. Stafford. Patterns of Enterprise Application Architecture, Addison Wesley, 2002.

[7] C. Nock. Data Access Patterns: Database Interactions in Object-Oriented Applications, Addison Wesley, 2003.

[8] C# Persistence Layer Source Codes, available online at: http://Csharp-source.net/persistence.

[9] Java Persistence Layer Source Codes, available online at: http://Java-source.net/persistence.

[10] G. Reese. Java Database Best Practices, O'Reilly, 2003.

[11] Recommended Practice for Architectural Description of Software Intensive Systems. Technical Report IEEE P1471-2000, IEEE Standards Department, The Architecture Working Group of the Software Engineering Committee, 2000.

[12] P. Clements, R. Kazman, and M. Klein, Evaluating Software Architectures: Methods and Case Studies, Addison Wesley, 2002.

[13] Data Access Layer Simulator (DALSim) source code available online at: http://www.4shared.com/rar/E2Vzoa8U/DALSim-Source_Codes.html.

[14] G. Nejad HajAli Irani, V. Tawosi, MVSDAP: a new extensible, modifiable and secure data access pattern for layered Information Systems, International Journal of Computer Science Issues, Vol. 9, Issue 2, No 1, 76-84, 2012.

[15] Stored Procedure Based simulation source codes available online at: http://www.4shared.com/rar/V9fW0Pri/DAP2-DAP2SP.html.