An Approach to Evaluation of PSM/MDA Database Models in the Context of Transaction Performance

Iwona Dubielewicz[†], *Bogumila Hnatkowska*[†], *Zbigniew Huzar*[†], *Lech Tuzinkiewicz*[†],

Wroclaw University of Technology, Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland

Summary
The aim of the paper is to propose a method for evaluation of logical database models in the context of transaction performance. Logical database models can be regarded as PSM models within MDA architecture. The logical database models are driven from a given conceptual database model (a part of PIM model within MDA) by applying different transformations rules. The evaluation method takes into account both static and dynamic aspects of logical database models. Within static-based aspect structural metrics to logical database model evaluation are considered. Within dynamic-based aspect transaction complexity is approximated.

Key words:
MDA, PSM, database models, performance evaluation

Introduction

Developing an information system usually entails designing a database. During this process, assuming a systematic approach, several models are developed at various levels of abstraction, preserving domain integrity, and intra- and inter-model consistency. The process can be divided into three basic phases: conceptual, logical and physical modelling [1,2,9,10,13]. The conceptual model, describing the application domain, yields the logical model specification, basing on which the physical model (implementation of the database) can be established. The structure of the process mandates the use of the MDA (Model Driven Architecture) approach for database design. MDA is an initiative by the Object Management Group to automate generation of platform-specific models from platform-independent models. MDA's approach to software development bases on modelling and model transformations. The process of software development is seen as modelling, i.e. constructing a sequence of models, starting with an application domain model and ending with a system implementation model. Each model, except the first one, is derived from its predecessor by means of transformations, stepping down from higher abstract models to less abstract ones, more closely aligned with the actual deployment platform. The final result of this process is a system code model. MDA does not enforce any particular software development methodology. There are three categories of models in MDA: Computation Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM).
CIM is a view of a system from the computation independent viewpoint. CIM does not show details of the structure of system. CIM may be identified with a domain or business model in other methodologies [6,12]. It describes both the requirements for the system and the environment in which the system will be used.
In our presentation, PIM and PSM are the main MDA's models of interest. PIM is a model that is independent of the features of a platform that the system may use. PSM is a model that is specific with respect to some features of a platform. PSM models may be called implementation models, and may be considered on different abstraction levels. Eventual implementation model bases on a given platform, which is understood as a set of subsystems and technologies that provide a coherent set of functionalities through interfaces and specified usage patterns.
In the paper, the conceptual database model is treated as a part of PIM. The logical model and physical models are treated as parts of PSM models [6]. For specification of PIM models, the UML 2.0 standard is recommended [7,16]. PSM models may – on the technological level – be expressed in SQL 99/2003 [2,4,5,11], while on the implementation (deployment) level they take the form of tool native scripts.
PSM is a platform-specific model; hence for databases it requires selecting both: the relevant technology (PSM_1), i.e. relational, object-oriented or XML, and a database management system (DBMS) supporting a given platform (PSM_2), i.e. Oracle, MS SQL 2005. We assume that all DBMSs taken into consideration are compliant with established SQL and XML standards [11].
One of the first stages of software system design is the specification of requirements, i.e. properties or characteristics that the system or its components must exhibit [7, 14]. We distinguish between functional and non-functional requirements. Non-functional requirements can be implemented in various ways. For systems containing databases, some of those requirements, for instance those relating to reliability or efficiency, can be delegated to the database

Manuscript received October 5, 2006.
Manuscript revised October 25, 2006.

* The work is supported by polish MNiSW within research grant 3 T11C 06430

management system. Decisions regarding the delegation of responsibility for the fulfilment of non-functional requirements affect the database development process. So, the quality and particularly effectiveness of those systems depend on the databases and on the way they provide information services. Hence, while designing an information system we should consider database design both in its functional and non-functional aspects. We should be aware of the many issues, but, in the paper, we concentrate on performance, i.e. data processing efficiency.

From the point of view of database design, functional and non-functional requirements, together with the domain model, and the set of business rules, constitute a CIM model.

The database developing process requires a sequence of transformations fulfilling the following principles:

- Elements from a target model should be traced from respective elements from the source model.
- Transformations should be conservative, i.e. no model property can be lost by a transformation.
- Transformations should guarantee correctness of a target model provided the source model was correct.
- For each model element at a given level (CIM, PIM, PSM_1) at least one transformation into an element of a more refined model must exist.
- Each model could be refactored.

PIM database model must contain all elements necessary for the satisfaction of functional requirements as well as non-functional ones. By treating functional and non-functional requirements as elements of the CIM model, we assume that transformations of data models ensure the required range of computing resources and thus enable to implement the required system functionality. This assumption is acceptable provided that the model transformation rules are unambiguous and non-contradictory. In more details, the PIM consists of:

- a class diagram with possible constraints,
- an estimation of maximum number of class instances,
- a set of transactions on the database and frequency of their calls.

The first two elements characterize system's requirements; the last characterizes system's environment. For the sake of simplicity we do not consider collisions among transactions. PSM consists of:

- a set of interconnected tables,
- a set of transactions expressed in terms of SQL statements.

Note that the set of transactions in PSM is a result of transformation of the set of transactions from PIM. Obviously, transformation of these transactions depends on the transformation of a conceptual database model on PIM level into logical model on PSM level.

There are many ways of model transformation, i.e. a given source model may be transformed into many target models. Typical transformations should yield a set of interconnected tables in third normal form. There are also other transformations, producing sets of non-normalised tables, which usually have other important features, especially that are more efficient in use.

So, the questions arise: how to evaluate the target models to select the best of them, and how to set up respective transformation strategy. In the paper, we address only the first question. We concentrate on transformation of PIM into PSM database model. Additionally, we assume that performance of transactions will be the criterion for the best model selection.

The aim of the paper is to propose evaluation method of PSM database models that are result of transformations of PIM database model in the context of transaction performance. The evaluation method takes into account static-based and dynamic-based aspects. Within static-based aspect structural metrics for strategy evaluation are considered. Within dynamic-based aspect transaction complexity is estimated.

Section 2 presents two approaches, i.e. static, and dynamic, to PSM models evaluation. The metrics associated with both approaches are presented in subsections 2.1, and 2.2 respectively. An example of using proposed metrics to PSM models assessment is shown in Section 3. Section 4 gathers some concluding remarks.

2. PSM models evaluation

When designing a database it is required that developed data model describes as far as possible the real situation being under investigation. It means that the determined data as well as relations among them result from existing business rules and domain constraints.

A logical data model, understood as a relational data model, can be obtained as an outcome of application of any methodology, taking into consideration different selection criteria for data structures as well as for data processing e.g. elimination of redundant data, elimination of anomalies connecting with data modification, ensuring of acceptable level of data performance etc. Generally, there is obtained a set of different solutions (data models). The decision which model to choose is based among others on subjective experience of the database designers. Consequently, a physical model often is a subject to modification for improving some of particular features of database e.g. increasing of processing performance.

The metrics presented in subsequent part are related to performance aspect of database and they enable to evaluate data model while logical modeling is conducted (at PSM_1 level).

2.1 Static approach

Data processing improvements in database-oriented systems is a well-known issue. In many research papers, e.g. [3,9] there are given practical recommendations which can help in tailoring of data model for users' requirements. However, these proposed processes are very tedious and time-consuming as they are most often implemented as multiple data model transformations which still need to be thoroughly tested. The test results constitute a basis for the possible data model acceptance.
For that reason we think that metrics which allow for an in-advance evaluation of some data model features would be much more valuable.
Hence we propose some metrics which has been elaborated on the basis of the authors' experiences in implementation of many different database-oriented information systems.
Taking into account the fact that relational database systems are for the time being the most important ones we focus only on this kind of data models. Proposed metrics are supposed to apply to relational databases and relate to the issue of performance.
Database performance in database systems hinges on many factors. We have restricted our research only to data models. We have omitted the factors associated with the environment in which the system is working (hardware, operating system). We have considered only those elements which can be taken into account at the PSM_1 model creation phase.
In order to obtain the best performance of database systems we should consider, during logical data modeling, the following important issues:

- kinds and complexity of database queries;
- frequency of transactions;
- the data structure;
- table schema;
- amount and kinds of constraints (domain, referential);
- estimated size of tables.

Elements mentioned above have been considered in the proposed metrics.
For given PSM model static complexity *MS* of a set *S* of transactions performed within a given period of time is calculated as the weighted sum of MS_T metric counted for every transaction:

$$MS_S = \sum_{T \in S} g_T MS_T$$

where:

g_T – frequency of transaction *T* within a given period of time;

MS_T – static complexity of transaction *T*.

Static complexity of transaction *T* is defined by MS_T metric:

$$MS_T = (\sum_{j \in Table_T} \left(c_{\mathrm{mod}} * d_j * s_j * c_{sel} + c_{ref} * \sum_{k \in FTable_j} fs_k \right) * c_{key}) * c_{join}$$

where:

$Table_T$ – set of tables affected by transaction *T*;
c_{mod} – cost of table modification (considered only when *T* is a modifying transaction);
d_j – record length factor for the *j* table; it is counted as: *maximal_record_length* div *page_length* + 1 (*page_length* is assumed to be equal to 1024);
s_j – maximal number of record instances in *j* table;
c_{sel} – cost of conditional clause in transaction *T* (e.g. *where* clause);
c_{ref} – cost of reference integrity assuring;
$FTable_j$ – set of tables for which referential integrity should be considered;
fs_k – maximal number of record instances in *k* table;
c_{join} – cost of table joining;
c_{key} – correction factor if tables join is realized basing on primary tables keys.

The MS_S metric aims at evaluating the logical model of database. Therefore, we have not considered such database properties like, for instance, indexes or features of hardware of a chosen database system.

2.2 Dynamic approach

In dynamic approach performance of PSM models is evaluated basing on assessment of transactions complexity. The transaction complexity is a similar concept to computational complexity of classical algorithms. In our case transaction complexity is expressed as a formula yielding an absolute integer value that may be compared with others values.
At PSM level transactions are represented by a sequence of SQL statements. To evaluate complexity of SQL statement it is decomposed into a set of so called elementary operations. There are three kinds of elementary operations:

- algebraic elementary operations: projection (PROJ), selection (SEL), join (JOIN);
- data modification elementary operations: insert (INS), update (UPT), delete (DEL);
- reference integrity elementary operations: CHK.

For example, an SQL statement of the form:
DELETE FROM *table* where *condition*;
is decomposed into two different elementary operations: selection (SEL), and deletion (DEL). If the table is associated by a foreign key with other table(s), additionally

decomposition of this operation will result with CHK operation(s).
Decomposition of SQL statement into elementary operations at this moment is done manually. For each elementary operation a maximal number (pessimistic estimation) of affected records is determined. The record numbers are treated as the problem size – it is obvious that a time of operation performance is a function of records' count. In turn, a time of record performance depends on record's length and on the kind of performed operation.
Dynamic complexity of transaction *T* is defined by MD_T metric:

$$MD_T = \sum_{\alpha \in Op} c_\alpha \sum_{r \in d} n_{\alpha r} c_r$$

where:

Op – set of elementary operation types;
c_α – cost of operation of a given kind α;
$n_{\alpha r}$ – maximal number of records affected by elementary operations of type α working on records of a given type length r ($r \in D$);
c_r – cost of record performance of a given type length r ($r \in D$);
D – set of record type lengths (counted as: *maximal_record_length* div *page_length* + 1)

Dynamic complexity of a set *S* of transactions performed within a given period of time is defined by the MD_S metric:

$$MD_S = \sum_{T \in S} g_T MD_T$$

where:
g_T – a frequency of transaction *T*, $\sum_{T \in S} g_T = 1$.

3. Case study

This section presents an example of static and dynamic approaches to PSM models evaluation. PIM model is the start point of our considerations.

3.1 PIM, and PSM data models

As it was mention above, PIM data model is described by a class diagram. Exemplary class diagram is shown on Fig. 1.
Customer, and *Employee* classes are two specialisations of *Person* class. Each person has only one address, but a given address may be associated with many people. An employee is responsible for many customers, and a customer is serviced by one employee. Employees work in a firm. The firm is placed in one address.

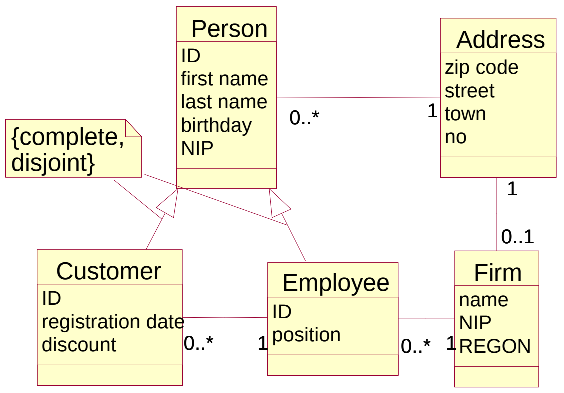


Fig. 1 Exemplary PIM data model.

PIM model is described with a set of important, in the context of database performance, features. For each class a maximal number of instances and maximal record length (in bytes) is approximated. Table 1 presents such assessments for two variants of database instances. The first variant was prepared for a medium-scale enterprise, while the second – for a small-scale enterprise. Record length factor is calculated basing on the approximations of maximal record lengths, as it was defined in subsection 2.1.

Table 1: PIM model characteristic

Class	Variant I	Variant II	Maximal record length [B]	Record length factor
	Max instance number	Max instance number		
Person	100050	3003	600	1
Address	80000	3000	400	1
Client	100000	3000	600	1
Employee	50	3	1400	2
Firm	1000	10	1030	2

Additionally, at PIM level we have informally defined system transactions and their characteristics. Transactions are traced from CIM model. At PIM level each transaction is described in terms of required classes, and forecasted frequency. Tables 2, and 3 present a set of five transactions for considered PIM model together with their characteristics. The characteristics (frequencies) are defined in two variants (for medium and small-scale enterprises).

Table 2: System transactions

No	Transaction	Required assets (classes)
1	Count the number of clients	Client
2	Count the number of clients above 65 years old	Person, Client
3	List the clients from a given town	Person, Client, Address
4	Insert an employee	Person, Employee, Address
5	Insert client	Person, Client, Address

Table 3: System transactions and their characteristics

Trans No	Variant I			Variant II		
	Freq	Freq/ Day	Norm. freq	Freq	Freq/ day	Norm. freq
1	1/day	1	0,0193	1/month	0,03	0,0059
2	1/month	0,03	0,0006	1/month	0	0,0000
3	1/month	0,03	0,0006	1/month	0,03	0,0059
4	5/week	0,71	0,0137	1/month	0,03	0,0059
5	50/day	50	0,9658	5/day	5	0,9823

Based on PIM data models, manually or automatically, different PSM models are elaborated. The models can differ with many characteristics, particularly they may have different performance characteristics.
Conceptual data model from fig. 1 could be transformed into many PSM models. Figures 2-5 present four possible versions of PIM transformations.
PSM models are expressed with terms of tables and foreign key references. A dependency with a <<fk>> stereotype means, that a source table has a foreign key to a target table, e.g. *Persons* table has a foreign key to *Addresses* table (see Fig. 2).
For the sake of simplicity the following naming conventions are assumed:

- a table containing instances of a given class is named with a plural form of the class name, e.g. *Person* class instances are gathered in *Persons* table;
- a table may contain instances of two or more classes; in such case its name consists of the names of all classes (in plural form).

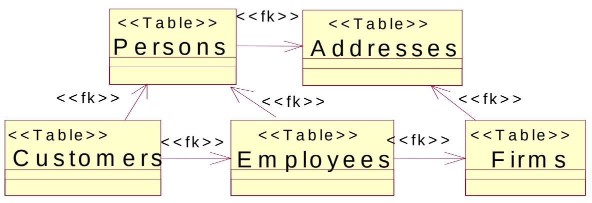


Fig. 2 PSM model – variant I

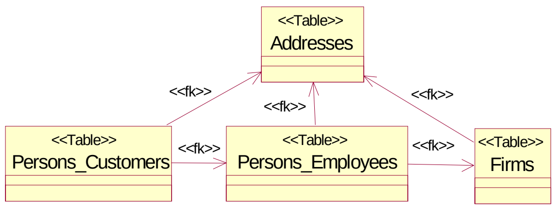


Fig. 3 PSM model – variant II

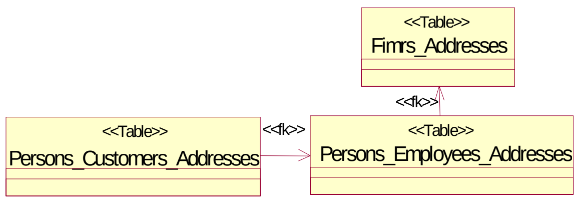


Fig. 4 PSM model – variant III

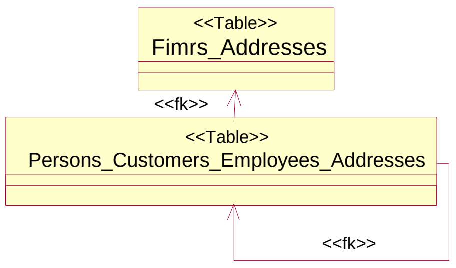
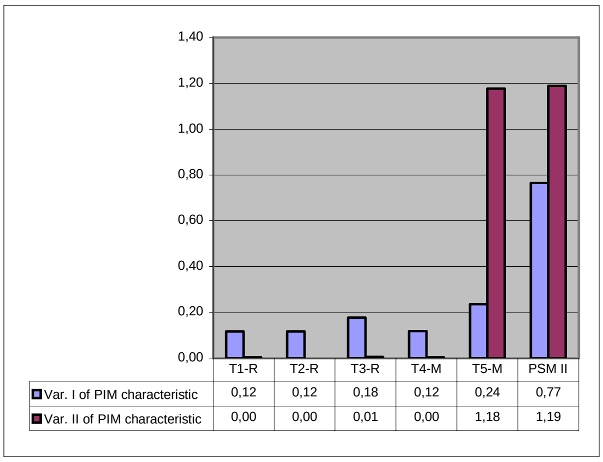


Fig. 5 PSM model – variant IV

3.2 Model evaluation – Static approach

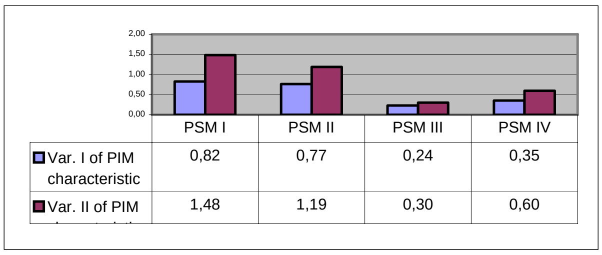
This section presents evaluation of static metrics, defined in subsection 2.1 for considered example. All four versions of PSM models were assessed by counting MS_T metric for 5 transactions, and for two variants of PIM model characteristics. The calculations were done with the assumption that: $c_{mod}=1$, $c_{sel}=1$, $c_{ref}=1$, $c_{join}=1$, $c_{key}=1$, $d_j=1$, and fs_k was equal to average value of records number in all tables.
Results of MS_T metric computation for PIM II are presented on the Fig. 6. The letter 'R' or 'M' added after the transaction identifier shows if the transaction modifies (M), or not (R) data in a table (tables).



	T1-R	T2-R	T3-R	T4-M	T5-M	PSM II
🟦 Var. I of PIM characteristic	0,12	0,12	0,18	0,12	0,24	0,77
🟥 Var. II of PIM characteristic	0,00	0,00	0,01	0,00	1,18	1,19

Fig. 6 MS_T metric evaluations for *T2* transaction

The final ranking of PSM models is presented on Fig. 7. The best is PSM III model. PSM IV is a little worse. The worst is the PSM – version I. The ranking is the same for both variants of PIM model characteristic, but the differences between models become lower when small enterprise is considered.



	PSM I	PSM II	PSM III	PSM IV
🟦 Var. I of PIM characteristic	0,82	0,77	0,24	0,35
🟥 Var. II of PIM	1,48	1,19	0,30	0,60

Fig. 7 Evaluation of MS_S metric for the set of transactions

3.3 Model evaluation – Dynamic approach

In order to evaluate the MD_T metric for a given transaction *T*, first the SQL statements representing this transaction on PSM model should be decomposed into elementary operations. Next, the maximal number of records, affected by each elementary operation must be assessed. Decompositions and assessments are here done manually basing on PIM model characteristic (see tables 1, 2, 3).

Exemplary SQL statement representing transaction: "Count the number of clients above 65 years old", and its decomposition into elementary operations for all variants of PSM models are shown in the tables 4 and 5.

Table 4: Input data for evaluation of MD_T metric for *T2* transaction

PSM	SQL Statement	Elem. operations	Max. no of records	Computational model	Comments
I	Select count(*) from join .... where year(date())-year(data)>65	JOIN SEL PROJ	N * M N N	N*M + 2*N	N – Max no of Clients M – Max no of Persons
II	Select count(*) from Persons_Clients where year(date())-year(data)>65	SEL PROJ	N N	2*N	N – Max no of Clients
III	Select count(*) from Persons_Clients_Addresses where year(date())-year(data)>65	SEL PROJ	N N	2*N	N – Max no of Clients
IV	Select count(*) from Persons_Clients_Employees_Addresses where is_klient and year(date())-year(data)>65	SEL PROJ	N M	N+M	N – Max no of Persons M – Max no of Clients

Table 5: Exemplary evaluation of MD_T metric for *T2* transaction

PSM	Computational model	Variant 1	Variant 2

I	N*M + 2*N	10005200000
		N = 100000
		M = 100050
			9015000
			N = 3000
			M = 3003

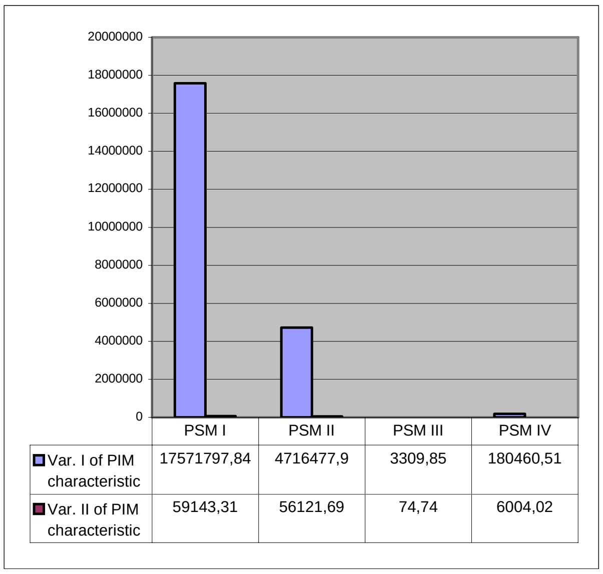
II	2*N	200000 N = 100000	6000 N = 3000
III	2*N	200000 N = 100000	6000 N = 3000

IV	N+M	200050
		N = 100000
		M = 100050
			6003
			N =3000
			M = 3003

Evaluations of MD_S metric for all transactions for each PSM variant are presented in the fig. 8. Evaluations were done for two variants of PIM model characteristic. The calculation was done with the assumption that c_α=1 (for each elementary operation kind), and c_r=1.

In both aspects – structural and dynamic, PSM variant III is the best. But let remember that PSI I and PSM II are in the 3rd normal form, while the others PSMs are denormalised. In that context ranking result is no surprise.

The same ranking of PSM models is for the second variant of PIM model characteristic, however in this case the differences between PSM models are lower. That is important in some cases, when not only efficiency, but also maintainability would be required feature of the database.



	PSM I	PSM II	PSM III	PSM IV
🟦 Var. I of PIM characteristic	17571797,84	4716477,9	3309,85	180460,51
🟥 Var. II of PIM characteristic	59143,31	56121,69	74,74	6004,02

Fig. 8 Evaluation of TB_S metric for the set of transactions

4. Conclusions

In the paper we have presented an approach to evaluation of performance of PSM database models that are developed within MDA. Performance of database is not an absolute value but it is considered in the context of its use – a set of transactions over the database. Two evaluation methods are presented and compared. The first static method is simple in use but it requires an expert knowledge. The knowledge relates values of factors that are used in the structural metrics. The second, dynamic method seems to be more complex as, except an expert knowledge, it requires decomposition of transactions into sets of elementary operations. The last problem would be avoided provided transformation rules for transactions were elaborated.

Simple case study shows that the methods are strongly related; both methods give the highest rank to the same PSM model, however, eventual conclusions need further experiments.

In general, our approach grows from the postulate to examine and evaluate developed models as soon as possible: in the case of database development, we suggest to evaluate logical models before elaboration of physical models. We have concentrated on performance evaluation, which is one of the most important non-functional characteristic of databases. Nevertheless, the similar approach could be adopted for evaluation of other non-functional characteristics. It would be reasonable to elaborate metrics related to other specific characteristics. A separate multi-criterion problem is how to integrate evaluation of individual characteristics into global evaluation of a system.

Future works require firstly exhaustive experiments to validate the proposed evaluation methods, especially to estimate ranges of factors that are used in our metrics. We expect that the relations among these factors will be not strongly depended on different platforms. Secondly, future works should bring automation of decomposition of SQL transactions into a set of elementary operations.

The next goal of our attempts would be to elaborate the strategies of PIM-PSM model transformations that yield the most efficient (in the sense of performance) PSM model.

References

[1] Ambler Scott W., Agile Database Techniques, Effective Strategies for the Agile Software Developer, John Wiley & Sons, 2004.
[2] ANSI/ISO/IEC International Standards (IS). Database Language SQL – Part 2: Foundation (SQL/Foundation, 1999, http://www.cse.iitb.ac.in/dbms/Data/Papers-Other/ SQL1999/ansi-iso-9075-2-1999.pdf.
[3] Connolly T., Begg C., Database Systems. A Practical Approach to Design, Implementation, and Management, Fourth Edition, Addison-Wesley, 2005.
[4] Database Languages – SQL, ISO/IEC 9075-*:2003.
[5] Date C., Darwen H., A Guide to the SQL Standard, Fourth Edition, Addison-Wesley, 1997.
[6] Dubielewicz I., Hnatkowska B., Huzar Z., Tuzinkiewicz L., Feasibility analysis of MDA-based database design, Proceedings of International Conference on Dependability of Computer Systems. DepCoS – RELCOMEX 2006. Szklarska Poręba. Los Alamitos, IEEE Computer Society Press, 2006.
[7] Fowler M., UML Distilled: A Brief Guide to the Standard Object Modeling Language, Third Edition, Pearson Education, 2004.
[8] Jacobson L., Booch G., Rumbaugh J., The Unified Software Development. Process, Addison-Wesley, 1999.
[9] Kiefer M., Bernstein A., Lewis P., Database Systems. An Application-Oriented Approach, Second Edition, Addison-Wesley, 2006.
[10] Marcos E., Vela B., Cavero M., A Methodological Approach for Object-Relational Database Design using UML, Software and Systems Modeling, Vol. 2, pp. 59-72, Springer-Verlag, 2003.
[11] Mattos N., Darwen H., Cotton P., Pistor P., Kulkarni K., Dessloch S., Zeidenstein K.: SQL99, SQL/MM, and SQLJ: An Overview of the SQL Standards, http://www.wiscorp.com/SQLStandards.html
[12] MDA Guide v. 1.0.1., http://www.omg.org/mda
[13] Naiburg E. J., Maksimchuk R. A., UML for Database Design, Addison-Wesley, 2001.
[14] Rumbaugh J., Jacobson L., Booch G., The Unified Modeling Language. Reference Manual, Second Edition, Addison-Wesley, 2005.
[15] Shasha D., Bonner P., Database Tuning. Principles, Experiments, and Troubleshooting Techniques, Elsevier Science, 2003.
[16] UML 2.0 Superstructure Specification, OMG, September 2003.

Iwona Dubielewicz received the M.S. and Ph.D. degrees in Computer Science from Electronic Department of Wroclaw Institute of Technology in 1972 and 1977, respectively. During 1984-199 she participated in designing and implementing NASK first computer wide-area network in Poland. Now she is a lecturer and her research area is dealing with software development methodologies and software project management.

Bogumiła Hnatkowska received the M.S., and PhD degrees in Computer Science from Wrocław University of Technology, Poland, in 1992 and 1998, respectively. Since 1998 she has working as a Professor Assistant in Institute of Applied Informatics at Wrocław University of Technology. Her main research interests are: software engineering, software metrics and models, software quality.

Zbigniew Huzar received the M.S., Ph.D., and habilitation degrees in Computer Science from Wrocław University of Technology, Poland, in 1969, 1974 and 1990, respectively. During 1978-1984, he was deputy director of Computer Center, during 1984-2003, he served as a head of Informatics Center, and since 2004 he has served as a director of the Institute of Applied Informatics, all at Wrocław University of Technology. The field of his interest comprises software engineering, especially formal specification and design methods.

Lech Tuzinkiewicz received the PhD degree in computer science from Wrocław University of Technology, Poland, in 1982. He works as a Professor Assistant in Institute of Applied Informatics at Wrocław University of Technology. His main research interests are: databases, data warehouses, software engineering.