DATABASE COMPLEXITY MEASURING METHOD

Mile Pavlić
Sveučilište u Rijeci, Odjel za informatiku, Adresa, 51000 Rijeka, Croatia
mile.pavlic@ris.hr

Marin Kaluža
Veleučilište u Rijeci, Trpimirova 2/V, 51000 Rijeka, Croatia
mkaluza@veleri.hr

Neven Vrček
Fakultet organizacije i informatike, Pavlinska 2, 42000 Varaždin, Croatia

Abstract:
There are so many methods on the market that could measure or estimate different aspects of an information system. When the information system is finished and it is in use for some time, redesigning of the existing one or making a new one has to be done. To make the estimation of investment as precise as possible, and to manage the development and implementation of the new information system, there is a need for measuring or estimation of the complexity of the existing information system. Information system complexity measuring methods show correct complexity measure, but their enforcement is very hard and slow. This paper represents a method called DC (Database Complexity). DC can obtain a measure of the database complexity very fast.

Key words: complexity, information system, measuring, database complexity, FP.

1. Introduction

Software development is an engineering discipline. The engineering approach implies that the planning and design phases precede the software development phase [18]. The engineering approach in software development allows for a project in which customer requests are collected and which develops the application respecting the "business rules". In the beginning of the software development project, very often, a software cost weight has to be estimated or measured. A logical consequence of the engineering approach toward software development is a need for introducing different types of measuring methods and quality estimation. We could measure the complexity of an information system (IS) [7], its productivity [14], cost [15], functionality [16][17], and so on. The measuring methods and metrics are constantly analyzed and upgraded to make the software measure or estimation as objective as possible [3].

A large number of software products were developed as a renewal (reengineering) of the existing one. Very often, there is a request for new software solutions on the market as a replacement or upgrade of existing (legacy) software. It is necessary to predict the real cost, expenses and development time of the new software. The cost and time depend on software complexity. The more complex the software is, the more time it will require to develop it.

This paper will present how software complexity could be estimated by measuring the complexity of different database concepts. Database concepts included in measuring are: attributes, keys, indices, database references – all used in software. This paper's objective is to present a method which could be used to make quick database complexity estimation.

The presented method will be applied on a business software that represent the IS (whole or partial) of an organization. The measured software is: car assurance, home assurance, fire assurance, etc. Specialized software will not be considered, for instance, multimedia software, software used as support to automated product processes, software with complex mathematical algorithms, and so on. Only the IS of a business organization will be analyzed [19].

2. Estimating and measuring methods

If there is an objective to develop whole software, there is also a very logical request: how could we measure software complexity. Software complexity could be measured or estimated by different methods. We have to emphasize, that essential factor in measuring or estimating process is, also, a software type. Software type defines its relation toward inputs and outputs, interaction with customer, customer interface, independent learning, and so on. In other words a software type defines its placement in three-dimensional vector space by Genetic taxonomy [1].

There are so many methods which could measure or estimate software complexity. Some of them are: Method of functional points analyses [4][7], Delphi [8], COCOMO [9], NVC [10], PND [5]. Some methods could measure, but some of them could only estimate software complexity.

Accepted referent method for measuring software complexity is method of functional points analyses (FP). The other methods are in generally reclined, or they are derived on FP. Propriety of some methods is proved by correlation with FP (PND [5]). Let's describe FP and PND methods.

2.1 Functional point analyses method

FP (Functional Point Analyses) method appeared by the ending of 70's. The objective was to give, as result, a number which will represent software complexity, and that number should be of importance. In other words, for two different software, a number of functional points could be given, and it could represent a real and objective difference between them. FP is a number without dimension defined in functional points that represents an effective relative measure of functional value delivered to the customer [2]. Regarding to the IFPUG, a FP model is consisted of transactional (EI, EO, EQ) and data (ILF, EIF) elements [3]. EI represents an External Input, EO External Output, and EQ represent an External Inquiry. ILF represent Internal Logical File, and EIF External Interface File.
There are defined rules, and by them some registered elements in the system are appended to transactional and/or data elements. The method is performed through the five steps, and gives a number of maladjusted functional points. After that follows relatively subjective calculating an adjusting value factor, and calculating adjusted functional points. The method is comprehensive, and it takes much effort and time to be performed [5].

To facilitate complexity measuring process, some methods for software complexity estimation has been created. Estimations are less correct, but measuring is faster and easier. Estimation methods could be proved by correlation level with FP (referent complexity measuring method). Estimation methods, in generally, could be categorized in two categories: direct estimation methods, and derived estimation methods [4]. Direct estimation methods are usually known as methods of expert's opinion and experience. Experts could directly estimate software complexity, but there is no mathematical evidence. Derived estimation methods gives complexity estimation like function of some variables which refer on some project attributes [5].

2.2 PND

Data on documents (PND) is a method which could be used for IS designing complexity estimation [5]. A method is performed in ten steps. All system documents have to be collected, and than continues computation of relevant data on each document. Summation of different data types gives a number which represents a system complexity. This method statistically correlates with referent measuring method FP [7]. PND could very fast accomplish system complexity estimation, and foresee measuring results by using FP method.

3. DC method

This paper defines a method called "**D**atabase **C**omplexity" (shortly DC). DC method could be used for database complexity measuring. Every business IS is composed of database and software. DC doesn't measure IS complexity, but it only estimate it. When measuring IS complexity, all its composed elements have to be measured. IS is composed of: hardware components, software, orgware, lifeware and netware. DC measures only logical structure of physical database used in its IS. A size of database itself will not be performed in measuring process. DC can, from data point of view, estimate software complexity. By measuring database complexity, DC method can foresee software complexity, and also IS complexity.

Physical database development is only one step within IS designing and developing methods [18]. Before database construction, there have to be made at least two different data models which represents some data relations. These models are entity-relationship (ER) model [11], and relational model (RM) [12]. ER represents appearance types on semantic level, with their properties (attributes), and their each other connections [11]. RM could be made by using some defined translation rules from ER model [18]. RM, beside attributes, also represents keys, and foreign keys in each relation.

Mentioned models will ensure as less (minimum) redundancy as possible in future database. If relation has some redundant attributes, there are anomalies also present. Anomalies could be occurred in insert, update and delete processes with tuples [13].

If IS has database modeled and normalized to minimally 3NF, DC method could easily estimate an IS complexity. Un-normalized relations in database, generally un-normalized databases, could not be measured and will not be measured by DC.

3.1 Method definition

Let's define basic measurement elements. To make calculation of database complexity, all relations in database have to be counted. For each relation all relevant elements have to be counted, and given number represents relation complexity, or relation weight **W**. Let **A** be a number of counted attributes in relation. Let **K** be a number of keys in relation. Key sum represent counting of primary and secondary keys. Let **I** be a number of counted indexes in relation. I imply only a number of un-unique indexes

in relation. Unique indexes were still counted in K. Let **F** be a number of foreign keys in relation. Weight W of each relation is a sum of counted elements, and could be shown by formula (1).

$$W = A + K + I + F \qquad (1)$$

Respecting a formula (1), W (weight-complexity) of each relation could be performed. To calculate complexity **C** for whole database, each relation W has to be summed. This is shown by formula (2). In formula, **n** represents a number of relations in database. W represents each relation weight, and C represents database complexity. **i** is variable that vary from 1 to n.

$$C = \sum_{i=1}^{n} W_i \qquad (2)$$

For every relation in database worth that each relation weight W is equal to sum of counted attributes A, keys K, counted indexes I, and counted foreign keys F.

W is relation weight, because it represents its complicated development and exploitation. This requires some labor which is in correlation with elements quantity, that database is consisted of. Each relation obtains its weight. Based on obtained weight, relations could be compared. Every single weight contributes in altogether database complexity.

4. Method enforcement

Let's define method appliance steps. DC method, on database of its IS, has to be enforced in eight steps:

1. Relation selection in database
2. Attributes counting – A
3. Keys counting – K
4. Indexes counting – I
5. Foreign keys (references) counting – F
6. Calculating total weight W – by using a formula (1)
7. Steps 1-6 perform on each database relation
8. Weight W summary of all counted relations – by using a formula (2)

Step 1. implies relation selection in database and its marking and preparing for next steps. Before this step it is possible to sort relations in database by particular argument (e.g relation's name). This will facilitate controlling whether each relation is treated by DC method.

Step 2. implies all attributes counting in relation. This also includes those attributes that are in database construction added, and represent unique identifiers so called ID attributes or surrogate keys [6]. Each attribute has its own semantic value. Even those attributes that are by different modifications (ID modification) added in database, also carry some semantic value. As system complexity directly depends on a number of stored and easy to use attributes, because every data could give information or at least a part of information, certainly each attribute has its own informative value 1. It isn't necessary to make a different scoring (evaluation) for every single attribute. Each attribute has equal importance to all others, and it carries equal quantity of information, and this is 1. So, by counting attributes, the attributes quantity A will be obtained.

Step 3. Keys counting – a primary key, and all secondary key will be counted. Relation's secondary keys are defined like unique indexes. Primary key and all secondary keys are preserving uniqueness of attribute values that are made of. Keys preserve un-repetition of some attribute's values in different relation's tuples. Duplication impossibility of particular appearance is preserved by keys. Key also has some semantic value. When looking at appearance (tuple) in relation, then key semantic value could be 1. Every appearance in relation could be managed in relation only once. By counting keys, relation key quantity will be obtained.

Step 4. Indexes counting – all relation's un-unique identifiers (indexes) are been counted. Those indexes that are created on database because of future foreign keys are not counted. Indexes are used on database for accelerating answers on sent queries. Indexes are incorporated in queries that are used in IS's software. Indexes do not have any semantic value like two types (attribute, key) mentioned above, but they accelerate more complex system's work. A need for using indexes could be later pointed out. Index and its usage arises system complexity. As indexes are used in a different places in a system, and probably many times, it is necessary to make evaluation on them when estimating IS complexity. Indexes do not have any semantic and information weight, but they allow faster accessing to the needed information. Every index should have weighting 1. By counting indexes, the index quantity I will be obtained.

Step 5. Counting foreign keys (references) – all foreign keys in relation have to be counted. Foreign key ensures that the tuple in relation A on attribute(s) that represent foreign key FK, will have a value which is additionally described and defined in another, by foreign key, connected relation B. FK ensures that some appearance (tuple in relation) in A will be described with predefined values (appearance – tuple in second relation) in B. FK also carries a semantic value. A tuple in A which has FK is additionally described by tuple (other attributes) from relation B. So, a tuple in A could be described by using reference on the other tuple. Using a FK allows obtaining "enhanced", but predefined information from B of

about tuples in A. So, every FK should also have a weighting 1. By counting foreign keys, the foreign key quantity F will be obtained.

Step 6. Calculating a total relation weight – a summation of values obtained from last 4 steps (steps 2-5). Results A, K, I and F are of equal importance. A is not more important than I, and F is not more important than K, neither is vice. Every single result carries equal weight to the others. If the results could be of different importance, when calculation is performed, some results should have participation coefficients in relation weight W. All counted results (A, K, I, F) have equal coefficient in calculation of W. So the weighting is 1. A W will be obtained, by using a summation operator over obtained results.

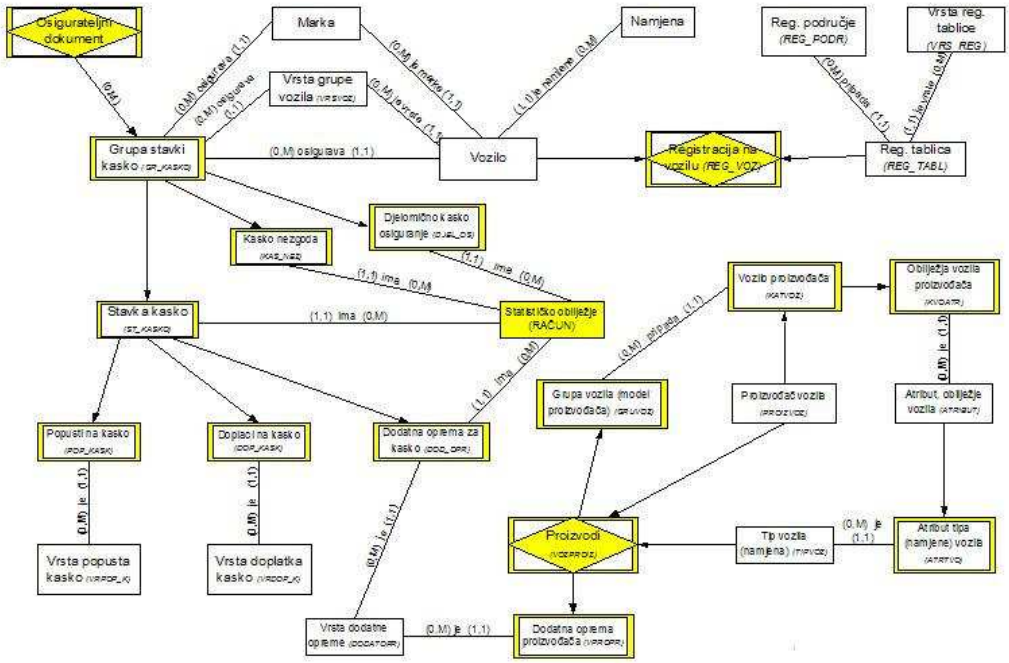
Step 7. Steps 1-6 have to be performed on every relation in database. Database also has, so called, system relations. Such relations are used by software for some user customization on database views. Such relations could store some system settings for software which is using the database. So in generally, relations that exist on database, and if they are not created from data model, they do not have any semantic value. Such relations could be called "Semantic Ballast". Such relations are not included in counting process.

Step 8. Every relation weight summation obtains total relations weight. It is not necessary to give different coefficients to each relation's weight. Every relation, depending on its weight, has some percentage participation into total system weight. Absolute relation weight shows it informational importance. Relations, generally, do not have equal informational importance. Because of that, their weights are distinct. Each relation weight will be as high as the summation of possible "information" is, carried by attributes, keys, indexes, and foreign keys. This clearly makes difference in information quantity carried by each relation. Summation of relations' weights will give a total system weight. This obtained weight could represent a measure for information quantity and technical operations by which the information could be quickly reached. Total system weight, should represent a measure of system complexity.

5. Method application results

Measuring with SBP method has been performed on 9 (nine) projects. Every single project has its own software and represents IS, or its part, in companies business. Measured projects are: glass breakage, earthquake, fire stocks, machinery breakage, household, burglary and robbery, fire summed/contracted, car all-risk insurance, car insurance.

Hereafter, there is a part of ER model of one system measured by SBP method. A model is part of the original project documentation [20]. Picture 1 represents a part of ER model of Car all-risk insurance system.



Picture 1 - ER model part – System Car all-risk insurance

System represented by Picture 1 is measured by DC method. Table 1 represents measured system results.

Table 1 – Relations of system – Carr all-risk insurance

RELATION	A	K	I	F	W
IMS_ATRIBUT	8	1			9
IMS_ATRTVO	6	1		2	9
IMS_DJEL_OS	15	1		2	18
IMS_DOD_OPR	22	2		3	27
IMS_DODATOPR	6	1			7
IMS_DOP_KASK	10	1		2	13
IMS_GR_KASKO	29	2	2	5	38
IMS_GRUVOZ	8	3		2	13
IMS_KAS_NEZ	15	1		2	18
IMS_KATVOZ	14	2		2	18
IMS_KVOATR	9	1		2	12
IMS_MARKA	6	1	1		8
IMS_NAMJENA	6	1			7
IMS_OSIG_DOK	104	2	15	17	138
IMS_POP_KASK	10	1		2	13
IMS_PROIZVOD	6	1			7
IMS_PROIZVOZ	6	1	1		8
IMS_REG_PODR	6	1			7
IMS_REG_TABL	8	1	1	1	11
IMS_REG_VOZ	10	2	2		14
IMS_ST_KASKO	41	2	1	3	47
IMS_STATIOBI	14	1	4	2	21
IMS_TIPVOZ	6	1			7
IMS_VOZILO	25	1	3	4	33
IMS_VOZPROIZ	7	2		2	11
IMS_VPROPR	9	1		2	12
IMS_VRDOP_K	7	1			8
IMS_VRPOP_K	7	1			8
IMS_VRS_REG	6	1			7
IMS_VRSVOZ	7	1			8
				Complexity-C	557

Table 1 represents measured values A, K, I, F, and results W and C for project Car all-risk insurance, whose model is showed on Picture 1. Table shows that A significantly affects on W. When A is bigger, W is also bigger. Other values K, I and F are significantly smaller and its effect on W value is also smaller.

Table 2 shows measuring results on 9 (nine) systems. In columns A, K, I and F are by order counted so called elements of DC method. Now column represent results from all counted relations in project. Column C now represents a summation on elements (A+K+I+F), or in other words, it represents a summation of each relation W.

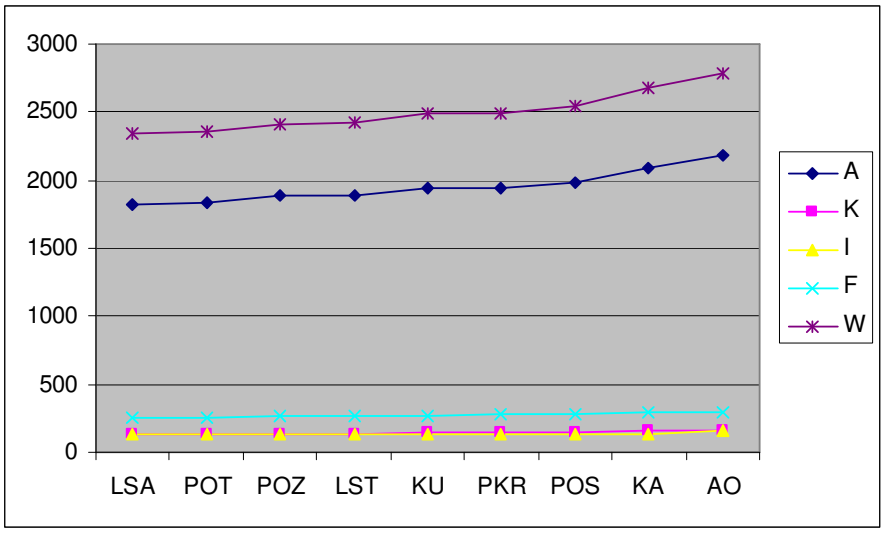
Table 2 – DC method measuring results

PROJECT	ACRONYM	A	K	I	F	C	C-A
Glass breakage	LSA	1822	133	128	259	2342	520
Earthquake	POT	1840	134	129	259	2362	522
Fire stocks	POZ	1883	138	128	265	2414	581
Machinery breakage	LST	1887	139	128	266	2420	533
Household	KU	1939	145	130	274	2488	549
Burglary and robbery	PKR	1945	145	130	275	2495	550

Fire summed/contracted	POS	1984	149	130	280	2543	559
Car all-risk insurance	KA	2090	164	139	289	2682	592
Car insurance	AO	2179	160	157	296	2792	613

Table 2 shows that Glass breakage system is the least complex system with attributes quantity 1822 and weight 2342. The most complex measured system is Car insurance with attributes quantity 2179 and weight 2792.

A graph of results is shown on Picture 2 and it represents relationship between counted elements A, K, I and F, and weight W. It is very easy to figure that a large part of weight is consisted of A. The other elements significantly less consist a W. In the other words could be said that an A proportionally correlates with K, I, and F. If number of relations in system is growing up, logically number of the other parts each, is also growing up.



Picture 2 – DC elements of measured systems

Picture 2 shows that amount of W is consisted of counted attributes. Number of K', I', and F' is significantly low. It is not good to say that this number is insignificant, because it carries some part of W, and by that affects total system complexity. As K, I, and F significantly less affect to total system complexity, and they significantly less carries a measure of total system complexity. Every mentioned element carries some semantic weight, and by that affects on total system complexity. In total system complexity measuring process, elements A, K, and F have equal weighting and it is 1.

Looking at a curve A and W, it could be concluded that values from a curve A could relatively precisely describe values on curve W. That has some implicated meaning, system weight and its complexity could be estimated, by counting only attributes in database.

Also could be concluded that relationship between value A and all other values (K, I, and F) is 3.5:1. That means that database has more than 3 times more attributes than all other elements together. There is a question: How it is in the other projects? If relation is the same or similar, then this relation could be rule in general.

6. Conclusion

This paper has represented a DC method which measures database complexity. The complexity can be easily measured. All available DBMS systems have the ability to get information from the data dictionary by standard SQL queries. If DBMS can get data dictionary information, then it can get all the required elements for the enforcement of the DC method. Hence, getting particular counted elements (A, K, I, F, and also W) could be very easily and quickly performed. Also, total database complexity C could be easily reached.

The database is used by its belonging IS. DC method could be used for fast IS complexity estimation. If the method is used for IS complexity estimation, DC method constraints have to be considered.

The measured results have shown that the number of attributes has a significant effect on database complexity. A number of keys K, un-unique indexes I, and foreign keys F, has significantly less effect.

Further research could be in a way to examine DC method applicability in other genetic taxonomic

arrays. Also, it could be shown that for calculating database complexity, there is no need to count keys, non-unique indexes, and foreign keys. Counting could be reduced to only counting attributes. Also, assigning a coefficient participation for each element (A, K, I and F), when calculating relation's weight W, could be considered. Relation's type could also give a coefficient, when calculating database complexity C.

We believe that the presented DC method could give a great contribution to the software development companies, and also to companies that are software consumers. To the first category, it could estimate how much it will take to finish the software development for the existing database, and plan the resources accordingly. To the second category, based on the existing IS parts and databases, it could estimate a cost for the other parts of IS, or the cost of software reengineering.

Bibliography

[1] Brumec, J., Vrček, N.: Genetic Taxonomy: the Theoretical Source for IS Modelling Methods, in Proceedings of the ISRM 2002 Conference, Las Vegas, NV, USA, 2002

[2] Abran, A., Robillard, P.N.: Identification of the structural weaknesses of Function Point metrics, 3rd Annual Oregon Workshop on Software Metrics, Portland, Oregon, 1991

[3] Beyers, C.P.: Estimating Software Development Projects, IT Measurement: Practical Advice from the Experts, IFPUG, Addison-Wesley, Boston, 2002, p. 337-362

[4] Meli, R. Santillo, L.: Function point estimation methods: A comparative overview, FESMA 99, Amsterdam, 1999

[5] Poščić, P.: Metoda procjene složenosti projektiranja poslovnih informacijskih sustava, Doktorski rad, Varaždin, 2007

[6] Kaluža, M., Čubranić, D.: Modifications on Data Model, IIS, Varaždin, 2007

[7] Garmus, D., Herron, D.: Function point analysis – measurement practices for successful software projects, Addison-Wesley, 2001

[8] Gordon, T. J.: The Delphi Method, http://www.futurovenezuela.org/_curso/5-delphi.pdf (pristupano dana 27.04.2008.)

[9] Gu, H., Tang. J., Shanmugasundaram, V.: Estimation of a Software Development project using COCOMO II, The Midwest Instruction and Computing Symposium, http://www.micsymposium.org/mics_2004/Gu.pdf (pristupano dana 27.04.2008)

[10] Ahm, S.I., Baker, S.: The NVC Method of Software Project Estimation, Jackson Open Kilobyte, Enterprises, http://www.tribalsmile.com/nvc/ (pristupano dana 27.04.2008.)

[11] Chen, P.P.: The Entity – Relationship Model – Towards a Unified View of Dana, ACM TODS, Vol. 1, No. 1, 1976

[12] Lyngbaek, P., Vianu, V.: Mapping a semantic database model to the relational model, ACM, SIGMOD, Vol. 16, Iss. 3, 1987, pp. 132 – 142

[13] Tkalac, S.: Relacijski model podataka, DRIP, Zagreb, 1993.

[14] Hamilton, S., L. Chervany, Norman: Evaluating Information System Effectiveness - Part I: Comparing Evaluation Approaches, MIS Quarterly, Vol. 5, No. 3, 1981, pp. 55-69

[15] Poppo, L., Zenger, T.: Testing alternative theories of the firm: transaction cost, knowledge-based, and measurement explanations for make-or-buy decisions in information services, Strategic Management Journal, Volume 19 Issue 9, 1998, pp. 853-877

[16] Parasuraman, A., Zeithaml, Valarie A., Berry, Leonard, L.: SERVQUAL: A multiple-item scale for measuring consumer perceptions of service quality, Journal of Retailing, Vol 64(1) 12-40, 1988

[17] Leyland, F. P., Richard, T. W., C. Bruce Kavan: Service Quality: A Measure of Information Systems Effectiveness, MIS Quarterly, Vol. 19, No. 2, 1995, pp. 173-187

[18] Pavlić, M.: Razvoj informacijskih sustava, Znak, Zagreb, 1996

[19] Jakupović, A., Pavlić, M.: The Meaning and Relationship of Relevant Elements in Business Organisation Structure, Cavtat, 2008

[20] Pavlić, M.:"IMIS", RIS, 2001.