SCIENTIFIC REVIEW

Effort assessment in the development of information systems projects

Zivadinovic Jovan[11], Medic Zorica[12], Markovic Blagojevic Marija[13]

Abstract

There is a great lack of methods and techniques in the software development process itself, as well as the lack of the appropriate tools that would make it more efficient. The significance of the problem is repeatedly emphasized by the need to ensure a high quality of software and software−based systems. The main objective of this work is to develop and systematize the original formal procedure for assessing the development of information systems in the early stages of the software life cycle, through metrics of the data model. We calculate the metrics of data model by using data that can be read off from a base data model, which is represented with an Entity−Relationship (ER) diagram that is defined with four basic concepts: entities, relationships, attributes of entities or relationships and values. The idea is to present the complexity of the process with a function of a number of these concepts and a number of attributes for entity types. Assessment techniques represent the basis for planning and successful performance of software projects. Statistical method was used in this paper and these assessment processes go under the category of empirical parametric methods, although they have some characteristics of the expert estimation method. A developed assessment process represents a step in the efforts to reach suitable measures which we would use to assess the size and complexity of the data model and also to estimate the amount of costs and resources necessary for the development of information systems. Likewise, certain metrics are developed. By being familiar with the data model, we can use these metrics to quantify characteristics of an information system as a whole in the logic design phase. Suggested metrics were tested on specific models and the results are shown here.

KEY WORDS: *software engineering, software metrics, costs assessment, data models*

JEL*: C87*

UDC: *005.8:004.41*

COBISS.SR-ID *219517708*

[11]Corresponding author Faculty of Business Economics and Entrepreneurship, Belgrade, Serbia, e- mail: zjovan50@gmail.com

[12] Graduate school of proffesional studies "Prof. dr Radomir Bojković", Kruševac, Serbia,e-mail:zmedic900@gmail.com

[13] Graduate school of proffesional studies "Prof. dr Radomir Bojković", Kruševac, Serbia,e-mail:mmarrija@gmail.com

Introduction

Early days of computer development defined software as a "work of art". What marked those early days was the existence of a severely small number of formal methods, and even smaller was the degree of their application. The software world was clearly undisciplined and the people who worked on software development really enjoyed that. Most of them still want to do that. However, the importance of software and software−based computer systems has reached such heights, that a proper and reliable software functioning has been imposed as a vital issue for the future of the human community. Today, software controls spaceflights as well as the nuclear potentials of "the great powers". Functioning of the market, payments and stock markets, banking and other similar systems are based on software. Global information services that are software−based are repressing other types of informing. Overall, we can expect that the importance of software will not cease to grow in the future.

Software engineering

A successful management over the software development and its use is in an immediate connection with a constant cost assessment and their reduction to a minimal amount necessary for a certain software quality. Acknowledging the effort and the costs is vital when it comes to the management of: profitability, feasibility of the project and productivity of the programmer. If we don't know how much effort and costs were necessary for previous projects of similar size and function, we cannot make a reasonable intuitive assessment of the forthcoming project (Bailey et al., 1981).

Certain researchers have dealt with the problem of cost assessment on the level of projects or macro level, while others have been oriented towards micro level, the level of programs themselves. What they have in common is that they express cost with the effort invested in designing, i.e. programming, with the inclusion of associated costs. Even though many models for effort and cost assessment have been developed, there is still no unique model suitable for all software development situations and for all development environments (Banker et al., 1994).

Metrics in software engineering

Basic problem in software engineering is not the lack of metrics; on the contrary, it seems that there are too many metrics, which are quite often used without fundamental understanding. The way things are, it looks like we are not coming up with solutions in this area. It looks like that during time the problem of measuring in the software area has become more present and more painful (Briand et al., 1999 p 313).

Reliability measure of a certain metric should show us how much trust we can have in certain results, depending on the variability of conditions in which the measuring is conducted. In other words, the question arises – to what extent does repeated measuring of the same phenomenon give the same results? Metric reliability can be brought to question depending on several types of changeability (Jeffery et al., 2000, p.109).

The most desirable metrics are the ones that represent a direct measurement of the observed characteristic. Unfortunately, those metrics are quite rare in the software world. There are often unavailable in times when they are needed the most, and often enough there is no way to perform a direct measuring. As a consequence, we limit ourselves and use indirect measures. When we say yes to using indirect measures, we become addicted to analytical or empirical model of the system or process we are measuring (Boehm et al., 2004, p.156).

Literature review

Halstead's "software science" theory is the first analytic software theory. Regardless of the fact that many scientists don't approve of the Halstead's "software science" laws, the importance of this theory is enormous, especially concerning the effort to try and define analytic equations and laws that would be applied to software systems. Halstead hypothesized that *language level* is a constant for a given programming language (Halstead, 1977).

De Marco pointed out that a functionality of a software system can be determined from the structural analysis components. For measuring purposes, De Marco classifies systems into three groups: function−based systems, data−based systems and hybrid systems. Functionality metric for function−based systems, called *Function Bang*, is calculated based on the complexity of DFD diagram and the type of operations on the given diagrams (De Marco, 1972).

Thomas McCabe suggested software complexity metric which is based on the program control graph. *Control graph* is a graph whose nodes are basic blocks (sequence of instructions without branching), and the branches represented a control flow (branching) of a program. The metric is defined as a cyclomatic complexity V(G) of a control graph of a certain module. One way to calculate V(G) is to define the number of regions in the planar control graph (T.J.McCabe, 1976).

Definitions

Bauer defined software engineering as: Establishment and use of sound engineering principles to obtain economically software that is reliable and works on real machines efficiently (Bauer, 1979).

Parnas defined software engineering as "constructing multi−version software by a larger group of people". This definition contains the essence of software engineering and it points out the difference between programming and software engineering: programming is primarily an individual activity, while software engineering is, in its core, a team activity (Parnas, 1978).

Data models

Data model represents an intellectual means that is used to describe statistical characteristics of the system, description of the characteristics in a certain stationary state (Lazarevic, 1993). Stationary state of a certain system is characterized by a set of dependencies which exist between the entities of a system. In a data model, these dependencies can either be presented with a data structure, or with a set of constraints on data values. Besides, it is necessary to define a set of data model operations, so as to use them in the process models to describe the dynamics of a real system. Data model enables us to interpret data on the observed existing system.

Metrics in data modeling

Data models represent a simplified notion of the relevant characteristics of a certain real system. This means that one real system can be presented by using several different data models depending on the overview and defined designer's goals. Bearing this in mind when it comes to defining data model metrics, we shall assume that they have to be applicable to any data model.

Further on in our research, in order to assess costs, i.e. time necessary to realize database on the basis of data model, we shall limit ourselves to the concepts of Entity−Relationship model and relational model and we shall try and define the metrics accordingly.

Metrics of data model structure

Data model structures, i.e. mutual connection of data, represent a base model of their interpretation. Structure in the ER model is defined with four concepts: entities, relationships, attributes of entities or relationships, and value. Starting from these basic concepts, we shall introduce base metrics and, based on them, we shall define the derived metrics, among which size and complexity of data model are especially important.

Considering that ER data model is usually presented with an ER diagram, and by conducting its analysis, we established base metrics which derive from basic concepts. In accordance with that, we defined the following base metrics:

- EJ the number of strong entity types,
- ES the number of weak entity types,
- EM the number of mixed entity types,
- V the number of relationship types,
- AEJ the number of attributes for strong entity types,
- AES the number of attributes for weak entity types,
- AEM the number of attributes for mixed entity types,

Value of base metrics is determined by simply counting the occurrence of certain concepts in the ER diagram. When ensuring transparency of the diagram, we should be careful not to repeat the same type of entities or relationships (Zivadinovic, 2000).

On the basis of base metrics, it is useful to define the following metrics:

– Total number of entity types in data model	$E = EJ + ES + EM$	(1)
– Total number of attributes for strong entity types	$AEJ = \sum_{i=1}^{EJ} AEJi$	(2)
– Average number of attributes for strong entity types	$\overline{AE}J = \frac{\sum_{i=1}^{EJ} AEJi}{EJ}$	(3)
– Total number of attributes for weak entity types	$AES = \sum_{i=1}^{ES} AESi$	(4)
– Average number of attributes for weak entity types	$\overline{AES} = \frac{\sum_{i=1}^{ES} AESi}{ES}$	(5)
– Total number of attributes for mixed entity types	$AEM = \sum_{i=1}^{EM} AEMi$	(6)
– Average number of attributes for mixed entity types	$\overline{AEM} = \frac{\sum_{i=1}^{EM} AEMi}{EM}$	(7)
– Total number of attributes in data model	$A = \sum_{i=1}^{EJ} AEJi + \sum_{i=1}^{ES} AESi + \sum_{i=1}^{EM} AEMi$	(8)
– Average number of attributes in data model	$\overline{A} = \frac{\sum_{i=1}^{EJ} AEJi + \sum_{i=1}^{ES} AESi + \sum_{i=1}^{EM} AEMi}{E}$	(9)

One important characteristic of the data model is its size. By using base metrics it can easily be defined as: *VM = E + V (10)*

When establishing size of the data model by using VM metric, the role of the concept in the data model is neglected, and we only observe its occurrence.

In order to estimate the amount of resources used in the realization of information system database, we shall introduce a metric called data model complexity. Data model complexity is a function of the number of entities, number of relationships and attributes. If we observe the global data model which contains types of entities, types of relationships and types of attributes, then we can express the complexity function SM as:

SM = f (E, V, A) (11)

If we simplify and take into consideration each of the concepts and then define certain weighted factors, data model complexity can then be expressed as a linear function:

$$SM = tj \times EJ + ts \times ES + tm \times EM + tv \times V + ta \times A \quad (12)$$

Based on our experience, value of the weighted factor, which is assigned to types of mixed entities, should be higher than the value of other weighted factors.

While defining the SM metrics, the cardinality of the relationships wasn't taken into consideration, yet it has a large impact on understanding the complexity of the system represented by the analyzed data model. Bearing this in mind, especially the influence of cardinality of relationships on preserving the integrity of the database, it is possible to

define a metric which will express the complexity of relationships. If we use the ER diagram making technique, i.e. types of relationships which, during the transition of ER model into relational model, become relations marked with mixed entities, then we can say that the equation (13) will in some way cover cardinality of relationships. It is clear that we can, by using in this way defined ER diagram, determine the number of BR relations in the model based on the values of the following metrics and that it is equal to the number of entity types, i.e.

$$BR = E \quad (13)$$

In practice, data model is usually presented through more than one sub−model. In most cases, sub−model covers one process. Use of the modeling methodology, with finding local sub−models, enables easier documentation, because making of ER diagram for one model is usually done on one A4 page.

If we assume that data model of a certain system is presented through sub−models, we can then, by analogy of the defined metrics, define metrics on the level of data sub−models (Zivadinovic et al., 2013):

- EJk number of strong entity types in k^{th} sub−model,
- ESk number of weak entity types in k^{th} sub−model,
- EMk number of mixed entity types in k^{th} sub−model,
- Vk number of relationship types in i^{th} sub−model
- m number of sub−model

Complexity of k^{th} sub−model:

$$SMk = tj \times EJk + ts \times ESk + tm \times EMk + tv \times Vk + ta \times Ak \quad (14)$$

Analysis of the practical application

We shall show the practical application of the introduced metrics by using the examples of local data sub−models of the information system for monitoring staff of the business system. Monitoring staff employed in education with eight sub−models was taken for this analysis from information subsystem:

- PM1 Employed;
- PM2 Personal data;
- PM3 Education;
- PM4 Stimulating measures;
- PM5 Housing security;
- PM6 Expert and scientific training;
- PM7 Work places and employing;
- PM8 Progress;

When calculating complexity of data sub−models, the weighted factors were assigned with the following values: tj = 3; ts = 1; tm = 4; tv = 2; ta = 0,5

These weighted factors were assigned with these values based on author's experience. Those values should be adjusted through practice, so as to gain the most realistic relations of the influence of certain values on complexity of data model.

Table 1: Values of entity and relationship metrics

Submodel number	Metrics			
	EJk	ESk	EMk	Vk
PM1	7	-	2	3
PM2	5	-	1	4
PM3	5	-	-	4
PM4	4	2	-	6
PM5	4	-	2	5
PM6	5	1	1	6
PM7	3	-	2	3
PM8	6	2	4	11
For the model	39	5	12	42

Source: Author's estimate

By going through these basic metrics, which are gained by listing all concepts from the ER diagram, we can assume that the PM8 sub−model is the largest and probably the most complex, because it contains the greatest number of strong, weak and mixed entity types and especially relationships. On the other hand, it looks like that sub−model PM7 for example has a small number of concepts and relationships, which could mean that this sub−model is small and not particularly complex in comparison to other sub−models. Given the fact that there are no metrics of the number of attributes among the metrics shown in table no. 1, and that we cannot see the influence of certain values on the complexity of the model, we can conclude that these base metrics can hardly be of use for the assessment of the sub−model complexity, i.e. data models.

Table 2:Values of metrics for the number of entity attributes

Submodel number	Metrics			
	AEJk	AESk	AEMk	Ak
PM1	24	-	3	27
PM2	13	-	1	14
PM3	16	-	-	16
PM4	14	4	-	18
PM5	15	-	4	19
PM6	15	2	2	19
PM7	11	-	8	19
PM8	14	6	7	27
For the model	122	12	25	159

Source: Author's estimate

Table 3: Values of the derived attribute metrics

Submodel number	Metrics			
	$\overline{AEJk}$	$\overline{AESk}$	$\overline{AEMk}$	$\overline{Ak}$
PM1	3.4	-	1.5	3
PM2	2.6	-	1	2.3
PM3	3.2	-	-	3.2
PM4	3.5	2	-	3.3
PM5	3.7	-	2	3.2
PM6	3.0	2	2	2.7
PM7	3.7	-	4	2.8
PM8	2.3	3	1.7	2.2

Source: Author's estimate

In table no. 2 and 3 we can see values of metrics that refer to attributes. Sub−models PM1 and PM8 have the most attributes, and in average per entity, PM4. If we observe only through attributes, sub−models PM1 and PM8 would be the largest and sub−model PM4 would be the most complex. This conclusion doesn't respond to our considerations on size and complexity of sub−models measured through entities and relationships.

Table 4: Values of derived metrics in data model

Submodel number	Metrics			
	Ek	VMk	RMk	SMk
PM1	9	12	39	48
PM2	6	10	24	34
PM3	5	9	25	31
PM4	6	12	32	36
PM5	6	11	30	39
PM6	7	13	32	41
PM7	5	8	22	30
PM8	12	23	50	71

Source: Author's estimate

We can see in table no. 4 that the largest sub−model (PM8) is also the most complex, and the smallest sub−model (PM7) is the least complex. However, the order of sub−models according to size and complexity is not entirely the same, which is a logical outcome. Namely, size of the sub−model takes into consideration only entities and relationships and neglects the attributes. Also, complexity metric takes into consideration that all components don't have an equal influence on the complexity of the model, while this is not the case with size.

It is particularly interesting to observe total number of entities because it is, as we mentioned before, equal to the number of relations in the relational data model which is gained from the observed ER model. It is considered that man/day number necessary for the making of database update application is equal to the number of relations. Therefore, we would need 12 man/day for PM8 and 5 man/day for PM7. Based on the data from table 4, we come to the conclusion that 1 man/day is approximately 6 SM. Of course, this is a

rough estimate and we should most definitely reach more precise connections between complexity metric and time necessary for the making.

In table 1, we can see values of ER base metrics on the level of data models and in table 2, we can see a metric of the number of entity attributes. Other metrics have the following values:

E=56, $\overline{AEJ}$ =3.1, $\overline{AES}$ =2.4, $\overline{AEM}$ =2, $\overline{A}$ =2.8, VM=98, RM=257, SM=333.

It is therefore evident that it is possible to quantify size and complexity of the data model. Knowing these values in the early stages of software development (these metrics can be calculated immediately after the making of ER diagram) represent a helping hand to the project leaders when estimating time distribution, costs and effort in the development of information system databases.

It is up to every individual organization to make a decision which of these metrics it will use in practice. However, it is important to stick to using the one metrics that we choose, and not to change metrics for each project (J. Zivadinovic et al. 2014).

Conclusion

In this paper, we made an effort in introducing metrics to data models. We defined metrics of size and complexity of the information system. Data models being the basic information system model, it is only logical to define metrics that refer to information system by using data model metric. Purpose of these metrics is to use them to perform a direct comparison of different information systems according to size and complexity, as well as to, in the early stage of information system development, get an impression about resources necessary for their realization.

Essential fault of the data model metrics defined in this paper is their insufficient verification in the engineering practice. Only after using these metrics in the making of different software projects can we talk about their validity. Also, by conducting verification on different specific models, we could find ways for further improvement of the suggested metrics.

References

[1] Bailey, J. W., Basili, V.R. (1981). A Meta-Model for Software Development Resource expenditures, Proceedings of the 5th International Conference on Software Engineering", pp 107-116.

[2] Banker, R. D., Chang, H., Kemerer, C. F. (1994). Evidence of Economies of Scale in Software Development, Information and Software Technology, vol 36 no 5, pp 275-282.

[3] Boehm, B. W., Brown R., Madachy, Yang Y. (2004). A software product line life cycle cost estimation model, in Proceedings of the International Symposium on Empirical Software Engineering , pp. 156–164.

[4] Bauer, C.D.(1979). Principles of Software Project Management, Adison-Wesley

[5] Briand, L. C., Emam K. El, Surmann D., Wiezczorek I.,Maxwell K. D. (1999). An assessment and comparison of common software cost estimation modeling techniques, in Proceedings of the International Conference on Software Engineering, pp. 313–323.

[6] De Marco, T. (1972). Controlling Software Projects, Yourdon Press, New York,.

[7] Halstead, M. (1977). Elements of Software Science, North Holland.

[8] Jeffery, D.R., M. Ruhe, I. Wieczorek. A, (2000). Comparative Study of Two Software Development Cost Modeling Techniques Using Multi-Organizational and Company-Specific Data. Information and Software , pp. 109-116.

[9] Kitchenham, B., Brereton, P. (2010). Problems Adopting Metrics from Other Disciplines, Workshop on Emerging Trends in Software Metrics, May pp 1-7.

[10] Lazarević, B. (1993). Structural System Analysis, Laboratory for Information Systems, Faculty of Organizational Sciences in Belgrade.

[11] McCabe, T.J. (1976). AComplexity measure”IEEE Trans.Software Eng.,vol.SE-2,No.12

[12] Živadinović, J. (2000). Metrics for software program quality assessment , SYMOPIS-2000, Book of Proceedings, Belgrade.

[13] Živadinović J., Medić Z., Jevtić B., Grozdanić R., Piljan I., (2013). The Software Projects Development Improvement by Applying Metrics; Metalurgia international vol. XVII, pp 149-158.

[14] Živadinović J., Medić Z., Ivković D, (2014). Assessment of expenses of software projects development; International Review No.3-4, pp 117-125

Article history:

- Received 1 September 2015
- Accepted 3 November 2015