Estimating the Complexity of IT Projects

Ion IVAN, Corneliu BĂRBULESCU, Gheorghe MATEI, Mihai Liviu DESPA
Bucharest University of Economic Studies, Romania
ivan@ase.ro, corneliu.barbulescu@ro.ibm.com, mgm1802@yahoo.com, despamihailiviu@gmail.com

Project Management practice leverage concepts and models that try to maximize assessment accuracy with regards to effort and resource consumption. A central aspect is analysis of project complexity. The higher the complexity, the higher the cost and resource usage and risk. In this article, we define project complexity classes and propose models for estimating those. Key factors that drive complexity have the utmost importance. Ultimately, guidelines on translating project complexity into cost and duration estimation are established.
Keywords*: Complexity, Models, Project Management, IT Projects, Costs, Estimates*
DOI: 10.24818/issn14531305/25.4.2021.01

1 Defining the concept of complexity

The objective of this article is to study the complexity of computer systems in order to build models that are used to increase the performance of measurement tools used during the implementation of projects.

Similar to a set, complexity should be considered a primary notion, and its clarification should be done using examples. Everything that is subjected to analysis must be first taken as a whole, consisting of interdependent components. Its components are either different or of the same type. The links between the components are more numerous or less numerous. The intensity of interactions is either more intense or less intense. Everything that is the subject of analysis must be assessed by structure and number of components, diversity of components, and links between components.

Projects in general and IT projects in particular are characterized by high complexity due to the following: high degree of unknown in terms of requirements for features that need to be implemented (requirements are most often incomplete), the diversity of activities involved in the implementation, the number of different types of raw materials used, the dynamics and intensity of the financial flows, high probability of risks occurring at all stages of development, the effects of decisions taken throughout the implementation cycle, reliance and dependency on subcontractors, and variations between what was planned and what was actually achieved.

Once we are able to fully comprehend complexity and once we are able to define it as a quality characteristic of the entity that's being analyzed, the next steps are: identify relevant factors that influence it, establish a set of exogenous variables, build models of complexity, and use of complexity in developing estimates.

All completed, ongoing or future projects are classified according to complexity in simple projects, medium complexity projects, high complexity projects and very high complexity projects.

If building a house is considered to be a simple project, building a four-story building is considered to be a medium complexity project, building a block of flats with related infrastructure is considered to be a project of high complexity, then the project aimed at the construction of an Olympic village with hotels, gyms, stadiums, swimming pools, athletics tracks and many other facilities, will be considered a project of very high complexity.

The same is true for IT projects. There are simple projects that consist of source code of several thousand lines, implemented by a team of no more than five programmers. Very complex projects are those carried out by teams of at least 100 high-class professionals, in which the number of modules exceeds the order of thousands and for the implementation of which advanced technologies are used and quality management is mandatory.

Computed complexity of the IT project is obtained by aggregating the data actually measured during the project life cycle.

Predicted complexity is obtained using specially constructed models in which it must be specified as input data, the number of types of resources, the volumes of activities, the acceptable level of risk, but also other variables. Hypotheses are constructed to which predicted levels of complexity are associated.

Planned complexity is the result of analyzing the existing resources and adjusting the objectives so that the planned resources required to carry out the project match as much as possible in terms of availability of the existing resources, so that the probability of stockouts is kept under control.

IT projects stand out from all other projects in that: the share of salaries in the total cost estimate is over 80%, high qualified workforce is required, reusing components is manageable to a high degree, the dynamics of the moral wear and tear of the technologies used are accelerated, and it all depends on the project management methodology and the way it's implemented.

There needs to be proper differentiation made based on when planning or measuring or predicting levels of complexity and what conclusions need to be drawn when there are significant differences between predicted levels and measured levels, and also the decisions that IT project managers need to make.

2. Factors of complexity

In the case of any entity whose complexity is the subject of analysis, components, functionalities and links between them need to be identified. The complexity of an entity depends on: how comprehensive the requirements are defined or how complete the design of that entity is, number of component types, number of components in each type, number of feature types, number of types of interactions, number of resource types, volume of resources, quality of resource, reliance on other projects, and the risks that were accounted for.

The complexity of IT projects is influenced by factors such as:

- NSI - the number of integrated services, by degrees of complexity. The number of integrated services result from activities such as the evaluation and identification, applicable especially in the case of IT integration projects (SOA). In practice, it is advisable to define an analysis stage to be carried out separately and a priori in relation to the execution project itself for which the complexity is analyzed. Each service is identified and assigned a degree of complexity depending on a number of features. (i.e. number of integrated applications, interface type - synchronous, asynchronous - number and type of processed messages, etc.)
- NAD - number of distinct activities. The number of distinct activities result from the project methodology applied to the project type. Thus, for a given project type, there will be an estimation model that includes a set of standard activities (Work Breakdown Structure (WBS)) to which a certain level of implementation effort will be associated depending on the complexity of the project itself.
- NR - number of roles. The estimation model includes – in addition to the standard set of activities – the roles that perform those activities, standard role and being of key importance in estimating the cost of project execution.
- NL - number of workers. Each role will be provided as 'consumed' by the execution of an activity in a given 'quantity', depending on the existing capacity for human resource. To simplify, for example, implementing the source code for a functionality provided with an estimated effort of 10 man-days can take 10 days for one developer and 5 days for two developers. Allocating a larger number of workers is usually done to accommodate more ambitious deadlines, but being conditioned by the availability of human resources (the availability of the required number of developers, in our example).
- EE - estimated effort. The effort is measured in man-hours or man-days and results directly from the estimation models developed in the case of an actual project, each activity being associated with an estimated effort depending on the

complexity of the activity. By dividing the effort associated with an activity (man-hours) by the number of workers (NL), the duration of that activity is determined. For example, dividing 10 man-days (EE) by 2 people (NL) the result is 5 days. Instead, by multiplying with the cost of one worker per day, the cost of performing that activity is determined.

- EM - measured effort. The calculation is similar, mathematically speaking, to the estimated effort (EE) with the distinction that it is measured after the completion of the activity in question.
- DE - estimated duration. Calculated as EE / NL, measured in days or hours.
- DM - measured duration. Calculated as EM / NL measured in days or hours.
- DM - measured duration of a project

If a project is viewed as a structured entity, the complexity study includes the number of components and the number of connections between components.

When the study of complexity needs to be taken further, diving deeper into details is required and as a consequence the following need to be defined: number of component types, the number of components of each type, number of link types, and the number of links in each type.

If the requirements are not fully known or available, the study of complexity may consider one or more scenarios, documenting a set of assumptions that may be validated or not, depending on what additional details emerge as more data becomes available with the progress of the project itself.

Complexity is usually studied for entities of different natures, because the problem of resource allocation is solved differently from one type of entity to another, but the comparability of complexities must be made and that is why it needs to consider those influencing factors that are found in each case.

3. Complexity models

It is interesting to construct models of complexity which ultimately by replacing the variables with measured numerical values for the studied entities, lead to obtaining numerical values, used in comparing entities, in accepting the levels of resource allocation and setting deadlines.

An IT project should be seen as a finished software product, consisting of components, often called modules, to which databases are attached.

A program consists of instructions. If the program or subprogram is regarded as a graph in which each instruction is associated with a node and the direction of execution of the instructions is associated with an arc, it will be noticed that there are programs or subprograms with graphs associated with several nodes and several arcs, respectively, there are graphs with fewer nodes and fewer arcs.

For lines of code whose instructions are executed one after the other, corresponding to a linear sequence, the associated graph has n nodes I_1, I_2, I_3, …, I_{m-1}, I_m, linked by n arcs, graph shown in Figure 1. In this scenario n = m - 1.

Fig. 1. Sequential code graph

If a three-tier Web application, in which the components are accessed sequentially, and the return values from any component is done only in the module of the root component, which is the R root node, we get as a result the graph depicted by the structure type shown in figure 2.

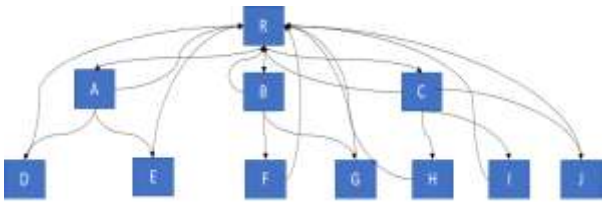

Fig. 2. Web application sample graph

On the second level we have the components A, B and C, and on the third level we have the components A, E, F, G, H, I and J. The Web application under consideration has 11 nodes, 10 arches that ensure the transition from the upper to the lower levels and another 10 arches that ensure the return from each node of the tree to the R root node.

In graph theory there is a C indicator, called the

cyclomatic number associated with the graph, defined by the formula [10]:

$$C = N - M + 2,$$

where N is the number of arcs in the graph, and M is the number of nodes in the graph.

In the particular case of the linear graph C = 1, which means that no matter how many nodes such a graph has, its cyclomatic number is the same.

In [10] the cyclomatic number corresponds to the complexity of a program, by means of its associated graph. Today's software projects are multi-component constructions, and the graphs associated with them highlight differences in complexity, especially because the desire to meet customer requirements to higher degree increases the number of connections between components, which in turns leads to a considerably increased number of arcs in the graph associated with the project.

In modern mathematics, information has been studied using quantitative methods, and the resulting indicators to measure entropy and diversification have led to the concept of software science and the complexity of the CS software, which is depicted by the formula [6]:

$$CS = f \log_2(f) + g \log_2(g),$$

where f is the number of different operands, and g - the number of different operators.

Programs written in different programming languages define operator lists that include, in addition to known operators, executable instructions. Operands can be counted either on definition or when they are used, the basic idea is for their number to be correct. There are programs that calculate the Halstead complexity of programs written in different programming languages. It is particularly important in establishing the hypothesis used in counting operands and operators to see if the complexities of some programs are comparable, calculating with various publicly available applications, which were provided as input with text files containing lines of code and which results in the CS complexities of the analyzed programs.

Starting from the idea that the executable instructions are different from each other and operands also differ, especially if dynamic memory allocation is considered, [1], [2], [3] two indicators were introduced:

$$CG = \sum_{h=1}^{k} f_h \log_2(f_h) \text{ and}$$

$$CR = CG/(h_1+h_2+\ldots h_k)\log_2(h_1+h_2+\ldots h_k)$$

These two indicators for measuring complexity are used for the indicators in table 1 in which data on several projects were collected. If we consider the following types of projects. Experience shows that IT projects that involved higher resource consumption also proved to be projects with higher levels of complexity.

Table 1. Categories of collected data related to IT projects

#	Project type	Standard activities	Complexity factors
#1	SOA DDI (SOA Design Development and Integration)	1. Solution Architecture 2. Macro design 3. Detailed Design 4. Unit Implementation and Testing 5. System Integration and Acceptance Testing 6. Post-Production Support (Hypercare)	a) Number of Services b) Complexity Distribution (Simple / Medium / Complex)
#2	Custom Development (AD)	1. Architecture Solution 2. Macro design 3. Detailed Design 4. Build / Release Cycles 5. Deploy 6. Hypercare	a) Use Cases number (User Stories) b) Complexity Distribution (Simple / Medium / Complex)

Today's IT projects requires a new approach to assessing complexity as there are no longer any boundaries in terms of measuring of how these projects are carried out, because the process of automatic data acquisitions increases the scope of the parameters that define, from a quantitative view, all the elements that contribute to the development of each project. Moreover, there are now tools that allow all measurements to be made on software and database products, but also data acquisition on the behavior of each component and how customers interact with the various components of computer systems, regardless of their

complexity.
The absence of information for the construction of complexity models is no longer invoked in any form. Same applies for their validation in order to be used effectively by developers, investors and especially by clients as users [16], [17].

4. Use of complexity models

If in the past, handling complexity in computer science was limited to software products, namely to source code, in the present the approach is completely different. IT projects fall into several categories, some tackle the development of new applications, others handle the integration of components and a third category deals with customization. For the three types of projects there are particularities that require differentiated approaches when it comes to resource allocation [4], [15].
Therefore, a model for IT projects complexity must consider factors such as: number of integration services, number of distinct activities, number of roles, number of workers, estimated effort, actual measured effort, estimated duration and measured duration.
There are several ways to analyze the importance of the influence of these factors, but factorial analysis in statistics, for which there is also a software package already implemented, provides sufficient information. It only takes a consistent set of data to get a ranking of the factors.
Assuming that the factors already listed are essential in the study of the complexity of IT projects, the model following model is submitted:

$$CPI = a_0 + a_1*NSI + a_2*NAD + a_3*NR + a_4*NL + a_5*EE + a_6*EM + a_7*DE + a_8*DM$$

The linear model associated with the complexity of IT projects is an aggregate indicator that has the sensitivity and non-catastrophic properties, but which is compensatory, which makes sometimes very different projects to have the same complexity.
Because of the fact that there are now databases that store information on the progress of many projects, for each class of projects a separate estimation of the coefficients can be undertaken as a_0 to a_8 and the complexity of a project is estimated using strictly the estimated coefficients of the class to which the project belongs.
When estimating the coefficients of the linear model of complexity, a table is built with the columns associated with the variables NSI, NAD, NR, NL, EE, EM, DE, DM, to which is added a measurement of CG complexity. If the CG Complexity is actually measured for projects already in use after they have been implemented, the CPI complexity will be used in making estimates to estimate the costs, time and effort required to carry out IT projects.
The only prerequisite is to record in the databases the historic data of each IT project, using techniques and methods to ensure data homogeneity and comparability [5], [12].
If over time there are too large differences between the estimated levels given by the implemented models and the levels recorded after the implementation of a project, it means that the techniques and methods of IT project development have evolved a lot, and the new projects implemented have less in common with the projects used in making estimates of the coefficients from a_0 to a_8 of the linear model. Consequently, a new set of data will be constructed using existing records in the databases, for the newly implemented IT projects and when using a statistical software product to estimate the coefficients of the linear regression model using the least squares method.
In [15] we have details on the construction of other model structures for estimating complexity, as well as their refining, in order to simplify the models as a number of operators and to reduce the number of variables as long as the models remain representative.
[13] and [14] present practical details on the construction, use and quality assessment of an aggregation model similar to the one presented in the current paper for measuring the aggregate complexity of IT projects.
Changing the paradigm of resource planning and allocation for the implementation on an IT project is of particular importance because the premises are created to compare the estimated levels of resources required for the project, with the level actually available or with the predicted

level either in a scenario of moderate optimism or in a pessimistic scenario. In the context in which a computer project today is considered an investment, the estimated complexity is necessary for the investor to see if he has the capacity to mobilize financial resources to finance the project, to the developer in order to see if he has the material and human resources to undertake the project and, last but not least, to users, to see if by accessing features they maximize their satisfaction in solving problems [8], [9] .

5. Conclusions

Complexity is a very important indicator in establishing the set of characteristics necessary in the evaluation of an IT project, because complexity influences the cost of the project, its duration, the size of the resources that are involved and the size of the risks to which investors, developers and even its users are exposed to.

Prospects for improving models for estimating and measuring complexity, the quality of the processes for estimating the coefficients of these models, are good, thanks to the efforts of all those that in the development of IT projects are involved in storing data about everything that happens throughout the implementation cycle. The existing tools allow the extraction from inhomogeneous databases of the essential indicators that define IT projects in order to compile the data sets necessary for the periodic estimates for the distinct types of IT projects, [11]. Moreover, using the algorithms to redefine the sub-intervals associated with the types of projects, updates will be obtained that will give a correct view of what a project of a certain class of complexity now means, compared to the exact same class of complexity a few years ago or a few decades ago. For example, a program with 300 lines of code in the COBOL language meant that it has medium complexity, and now a C# program with the same number of lines of code is a very low complexity program.

The complexity of IT projects is considered to be an open topic, scientific research still has many paths to follow, and the expected results are estimated to be among the most interesting and useful in project management in general and in IT project management in particular.

References

[1] Arhire, R., I. Ivan, M. Macesanu, "Program Complexity Analysis, Hierarchy, Classification," SIGPLAN Notices, vol 22, no. 4, 1984, pp. 94-102

[2] Arhire, R., Ion Ivan, M. Măceșaru, "Analiza comparată a complexităţii produselor program," in proc. of Conferinţa Naţională de Cibernetică - Cibernetica aplicată, 5-8 oct. 1983, ed. M. Florescu, and E. N. Mizil, Academiei, Bucharest, 1985, pp. 434-438

[3] Arhire, R., *Evaluarea complexitatii sistemelor de programe*, PhD Thesis, ASE Bucharest, 2000

[4] Ivan, I, Simion, F., Sinioros, P., Popescu, M., *Metrici software*, INFOREC, Bucharest, 1997

[5] I. Ivan, P. Pocatilu, D. Ungureanu, *Project Complexity*, INFOREC, Bucharest, 2001

[6] Halstead, M. H, *Elements of Software Science*, Elsevier Publishing, 1977

[7] I. Ivan, M. Popescu - Metrici software, BYTE Romania, vol. 2, no. 5, May 1996, pp. 73-82

[8] Karadimou, A., Ivan, I., A. Licuriceanu, G. Lupu, "Metrici de complexitate software bazate pe dependenţele instrucţiunilor," Informatica Economică, vol. 3, no. 3, 1999, pp. 11 - 19

[9] Măceșanu, M., Arhire, R., Ivan, I., "Clase de complexitate pentru produse-program," Buletinul Român de Informatică, no. 1/1985, pp. 63-68.

[10] McCabe, T.J. A., "Complexity Measure," IEEE Transaction of Software Engineering nr.2(4) 1976, pp. 308-320

[11] Milodin, D., Ivan, I., Georgescu M., "Project Complexity Evaluation," Journal of Information Systems & Operations Management - JISOM, vol. 4, no. 1, 2010, pp. 23 - 38

[12] I. Ivan, P. Pocatilu, D. Ungureanu, "Complexitatea proiectelor," Economie – teorie si practică, Supl. of Economistul, no. 229(831), 9 April 2001

[13] Tovissi L., Ivan, I., Arhire, R., Macesanu M., "Analiza comparativă a

modelelor de complexitate pentru produse program," Revista Română de Statistică, vol. 33, nr. 5, 1984, pp. 54-64.
[14] Tovissi L., Moscovici, E., Ivan I., "Utilizarea analizei entropice in studiul complexitatii programelor cu aplicatii la normarea activitatii de programare," in Proc. of "Creşterea eficienţei economice, componenta esenţială a unei noi calităţi a activităţii", 15-16 April 1982, ASE, Bucharest, 1983, pp. 293-298.
[15] Ivan, I., Vișoiu A., *Baza de modele economice*, ASE, Bucharest, 2005
[16] IBM: Keys to Building a Successful Enterprise Project Management Office; https://www.pmi.org/-/media/pmi/documents/public/pdf/white-papers/ibm-coe-whitepaper.pdf?v=e583a4eb-386b-4885-85e0-662555218cfe]
[17] Pomar E., IBM Whitepaper: 7 Keys to Successful Project Management, https://www.ibm.com/downloads/cas/J1OBPJZY]

Ion IVAN has graduated the Faculty of Economic Computation and Economic Cybernetics in 1970. He holds a PhD diploma in Economics from 1978. Currently he is professor of within the Department of Economic Informatics and Cybernetics at the Bucharest University of Economic Studies. He is the author of more than 26 books and over 80 journal articles in the field of software quality management, software metrics and informatics audit. His work focuses on the analysis of quality of software applications. From 1994 he is PhD coordinator in the field of Economic Informatics. His main interest fields are: software metrics, optimization of informatics applications, efficiency implementation analysis of the ethical codes in informatics field, software quality management and data quality management.

Corneliu BĂRBULESCU is Executive Architect with 20+ years outstanding track record of transforming businesses through IT Innovation, leading Strategy, Design and Delivery of Complex IT Systems. He is Member of IBM Academy of Technology and holds a distinguished certification portfolio featuring "Distinguished Architect/Chief Architect" from The Open Group, "Architecture Thought Leader" from IBM. Corneliu is a leader of Technical Community at IBM and is an IBM Inventor, working on patents and publications in Cloud and AI domains. He has a Bachelor Degree from Faculty of ECSI, Bucharest University of Economic Studies since 1997 and a Master's Degree in IT Project Management from the same institution since 2002.

Gheorghe MATEI has graduated the Faculty of Planning and Economic Cybernetics in 1978. He achieved the PhD in Economic Cybernetics and Statistics in 2009, with a thesis on Business Intelligence systems in the banking industry. He had a long career as a software developer in several companies, and then worked in the accounting and reporting department in Romanian Commercial Bank. His fields of interest include Business Intelligence systems, data warehousing, decision support systems, collaborative systems. He is a co-author of the books "*Business Intelligence Technology*" (2010) and "*The Digital National Wealth*" (2020), as well as author and co-author of several articles in journals, international databases and proceedings of national and international conferences in the mentioned domains.

Mihai Liviu DESPA has graduated from the Faculty of Cybernetics, Statistics, and Economic Informatics from the Bucharest University of Economic Studies in 2008. He has graduated a Master's Program in Project Management at the Faculty of Management from the Bucharest University of Economic Studies in 2010. He holds a PhD from the Economic Informatics Doctoral School. He is currently CTO at myOnvent. His main field of interest is the management of innovative software development projects.