The law of Demeter in IT project development based on design pattern

Kateryna Kolesnikova[1,†], Alexandra Klimanova[1,*,†] and Vadim Ziuziun[2,†]

[1]*International Information Technology University, 34/1 Manas St., Almaty, 050040, Kazakhstan*
[2]*Taras Shevchenko National University of Kyiv, 64/13 Volodymyrska St., Kyiv, 01601, Ukraine*

Abstract
The Law of Demeter outlines critical principles for structuring the interaction of software modules or patterns within a system, emphasizing the importance of minimizing dependencies to enhance modularity and maintainability. Initially designed for software development, this law has found broader applications in IT project management, mainly through its use in structural diagrams that reflect system hierarchies. Based on homomorphism, these diagrams ensure that project models accurately mirror the original structure, while Markov models help predict future project states and decision pathways. This paper explores the integration of Demeter's Law into project management frameworks, demonstrating that control schemes developed by this law exhibit simpler, more efficient, and more adaptable structures. By applying cognitive and Markov models, the study offers a method to quantify project progress and forecast outcomes, significantly improving the adaptability, scalability, and manageability of IT projects. Additionally, it highlights how these models can foster better decision-making and risk management, providing more reliable, data-driven insights.

Keywords
IT projects, management, structure, Demetri's law, communications, model, patterns

1. Introduction

The ever-changing field of IT project management is constantly questing for efficient methodologies. This has led to exploring different design patterns and logical approaches to software development. One such approach uses design patterns and reusable modules that act as "building blocks" to create complex software systems [1-6]. These patterns streamline the software design process, making it more efficient and adaptable. However, organizing these patterns presents challenges, particularly in maintaining a manageable and modular structure [7].

A key concept in addressing these challenges is the Law of Demeter (LoD) [8], which provides principles for organizing the interactions of software objects or modules. Initially applied in software development, the Law of Demeter has also been extended to project management. LoD principles simplify project systems' management, analysis, and adaptation when applied to project structural schemes. They provide a homomorphic reflection of the original project structure and predict future project states through Markov models.

This article explores the significant benefits of leveraging the Law of Demeter to optimize software development and project management. By applying its principles in the design of control schemes and project structures, you can enhance the efficiency and adaptability of systems.

DTESI 2024: 9th International Conference on Digital Technologies in Education, Science and Industry, October 16–17, 2024, Almaty, Kazakhstan
* Corresponding author.
† These authors contributed equally.
kkolesnikova@iitu.edu.kz (K. Kolesnikova)
0000-0002-9160-5982 (K. Kolesnikova)

2. Statement of the problem and the objective of the study

In a narrow sense, the law of Demeter defines the structure of interaction of software objects (modules) or patterns. In a broad sense, the effect of the law of Demeter also extends to structural schemes of project management, including those schemes that are included in GOSTs [7]. The objective of the study is to generalize the principles of constructing structural schemes of project management based on the homomorphism of the original (project) and its Markov model [7].

3. Literature review

The main problem with IT projects is that these projects are "black swans" — a term coined by Nassim Taleb [9], which describes rare, difficult-to-predict events with a huge impact. Another name for such projects is extreme projects, in which deadlines are critical, the cost of error is extremely high, and requirements change unpredictably, and the customer may decide at the last moment that he needs a completely different result [6].

Design patterns are one of the developer's tools that helps save time and implement design solutions at a higher quality level. Any design pattern is a formalized description of a frequently encountered design task, a solution to this task, as well as recommendations for applying the solution in various situations. In addition, a design pattern necessarily has a commonly used name [3, 7].

At the center of pattern technology is the idea of standardizing information about a typical problem and methods for solving it. In [3] it is shown that a pattern is a certain template, which was later called a form or format. The form uses five directions, according to which the discussion and application of patterns in specific situations was formalized.

An important stage when working with patterns is the correct modeling of the subject area under consideration. This is necessary both for formalizing the task at hand and for selecting appropriate design patterns.

Correct and timely use of design patterns gives the developer many advantages. A system model built in terms of design patterns is actually a structural representation of those elements and connections that are most interesting and necessary for solving the task [12]. In addition, a model built using design patterns is simpler and more visual. However, despite its simplicity and clarity, it allows for a deep and comprehensive study of the architecture of the software being developed. The use of design patterns increases the system's resistance to changing requirements and simplifies its inevitable subsequent revision, which reduces the impact of risk on the success of projects [5]. A set of design patterns, which, as a rule, have fractal properties [25], essentially represents a set of repeating elements and methods for solving unique project management problems.

4. Main part

The multitude of factors in weakly structured systems forms a complex "web" of connections and states that change over time. The development of projects in such a multifactorial system can only be represented in the form of qualitative models [26 – 31]. At the same time, the transformation of qualitative cognitive models into Markov models will allow us to move on to quantitative assessments of the progress and results of projects.

In cognitive modeling of complex processes, the key concept is cognitive map, which displays the structure of interaction of project processes as a directed weighted graph, in which [12]:

- the vertices correspond to the basic factors (states) of the project;
- direct connections between factors reflect cause-and-effect chains along which the influence of a certain factor on other factors is spread; it is believed that factors included in the "if...,

then...” condition influence the factors of the consequence of the entire chain, and this influence can be reinforcing (positive) or inhibiting (negative).

The cognitive map displays only the structure of connections between factors. It does not reflect the essence of the impact, nor the dynamics of changes in influences depending on changes in the situation or changes in the factors themselves over time. Displaying these features is possible at the next level of structuring information based on the cognitive model [13]. For this purpose, it is proposed to use the Markov model of state change, which allows displaying the multi-vector essence of random processes of project/program/project portfolio management.

According to SWEBOK [2], software development is based on the application of 10 main knowledge areas (Fig. 1):

1. Software requirements;
2. Design, software architecture;
3. Software design;
4. Testing;
5. Software support;
6. Software quality;
7. Tools and methods;
8. Software engineering processes;
9. Management in software engineering;
10. Configuration management.

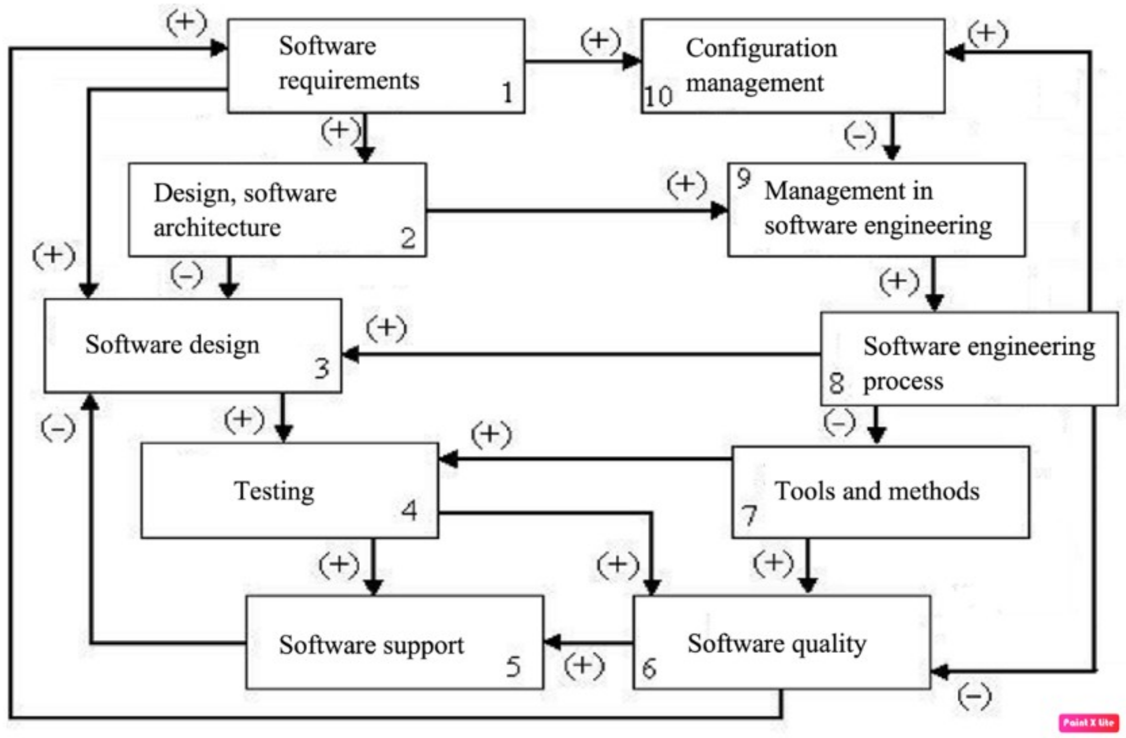


Figure 1: Cognitive map of software development.

The cognitive map of software development includes 10 nodes corresponding to the main areas of knowledge (competencies and connections between these nodes (Fig. 1). In fact, this cognitive map of software development displays the states of the system and the transitions between these states. If we assume that the sum of the probabilities of all states is equal to one, and also that the transitions from each state to other states are disjoint events, then such a graph can be represented as a homogeneous

Markov chain with discrete states and discrete time [10, 12]. To do this, we supplement the directed graph displaying the cognitive features of software development projects with delay connections in each of the 10 processes (states) and obtain a Markov chain. The matrix of conditional transition probabilities ||πij, i= 1 … 10; j =1 … 10|| for the Markov chain (Fig. 1) will have the form:

$\|\pi_{ij}\| =$

$\pi_{1.1}$	$\pi_{1.2}$	$\pi_{1.3}$	0	0	0	0	0	0	$\pi_{1.10}$
0	$\pi_{2.2}$	$\pi_{2.3}$	0	0	0	0	0	$\pi_{2.9}$	0
0	0	$\pi_{3.3}$	$\pi_{3.4}$	0	0	0	0	0	0
0	0	0	$\pi_{4.4}$	$\pi_{4.5}$	$\pi_{4.6}$	0	0	0	0
0	0	$\pi_{5.3}$	0	$\pi_{5.5}$		0	0	0	0
$\pi_{6.1}$	0	0	0	$\pi_{6.5}$	$\pi_{6.6}$	0	0	0	0
0	0	0	$\pi_{7.4}$	0	$\pi_{7.6}$	$\pi_{7.7}$	0	0	0
0	0	$\pi_{8.3}$	0	0	$\pi_{8.6}$	$\pi_{8.7}$	$\pi_{8.8}$	0	0
0	0	0	0	0	0	0	$\pi_{9.8}$	$\pi_{9.9}$	0
0	0	0	0	0	0	0	0	$\pi_{10.9}$	$\pi_{10.10}$

=

0.40	**0.10**	**0.20**	0	0	0	0	0	0	**0.30**
0	**0.65**	**0.15**	0	0	0	0	0	**0.20**	0,00
0	0	**0.60**	**0.40**	0	0	0	0	0	0
0	0	0	**0.50**	**0.10**	**0.40**	0	0	0	0
0	0	**0.20**	0	**0.80**	0	0	0	0	0
0.10	0	0	0	**0.20**	**0.70**	0	0	0	0
0	0	0	**0.10**	0	**0.40**	**0.50**	0	0	0
0	0	**0.20**	0	0	**0.20**	**0.30**	**0.30**	0	0
0	0	0	0	0	0	0	**0.10**	**0.90**	0
0	0	0	0	0	0	0	0	**0.10**	**0.90**

The specified transformation of the cognitive map into a Markov chain allows us to move from qualitative assessments of the progress of projects to quantitative characteristics. At the same time, quantitative assessments represent not only a multi-vector picture of the state of projects, but also have the property of forecasting [4].

The general solution for the Markov chain can be written in matrix form [5]:

$$\left\| \begin{matrix} p_1(k+1) \\ p_2(k+1) \\ \vdots \\ p_9(k+1) \\ p_{10}(k+1) \end{matrix} \right\|^T = \left\| \begin{matrix} p_1(k) \\ p_2(k) \\ \vdots \\ p_9(k) \\ p_{10}(k) \end{matrix} \right\|^T ¿ \left\| \begin{matrix} \pi_{1,1} & \pi_{1,2} & \cdots & 0 & \pi_{1,10} \\ 0 & \pi_{2,2} & \cdots & \pi_{2,9} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \pi_{9,9} & 0 \\ 0 & 0 & \cdots & \pi_{10,9} & \pi_{10,10} \end{matrix} \right\|$$

where

T is the transposition sign;

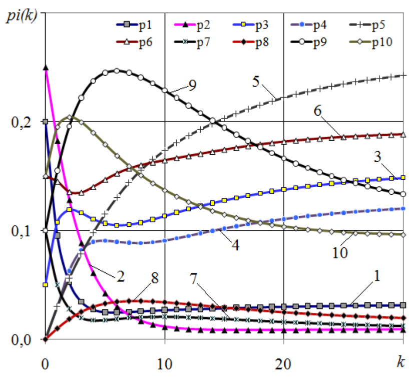
pi(k) and pi(k +1) are the probabilities of states at step k next step k+1;

i – state number.

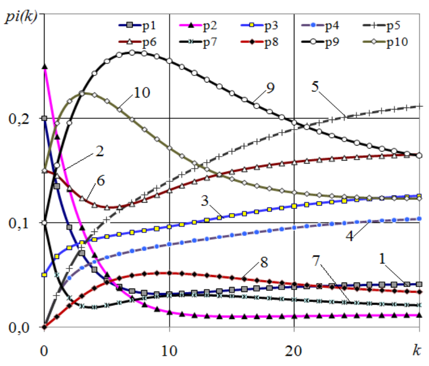
For the probabilities of states the following condition is true:

$$p1(k)+p2(k)+\ldots+p10(k)=1. \tag{1}$$

The obtained probabilities of states as a result of the actions performed allow us to predict and evaluate the effectiveness of project implementation. Fig. 2 shows an example of the results of modeling a system using a Markov chain. As can be seen, as the project is implemented, the degree of resource intensity of individual processes will change. For the accepted conditions of the basic version of the control scheme shown in Fig.1, the highest probability of the state after the 30th step corresponds to process 5 - "Software support" and process 6 - "Configuration management" (Fig. 2a). In this model, the specified processes act as patterns.



a) basic version of the circuit, π 1.3 = 0.2; π 8.6 = 0.2



b) modified scheme, π 1.3 = 0; π 8.6 = 0

Figure 2. Change in the probabilities of states of software development processes for different schemes: 1 - Software requirements; 2 - Design software, architecture; 3 - Software construction; 4 – Testing; 5 - Software support; 6 - Software quality; 7 - Tools and methods; 8- Software engineering processes; 9 - Management in software engineering; 10 - Configuration management.

A modified control scheme is to be constructed, taking into account the requirements of the Law of Demeter. As can be seen, communication along the graph arc between states (patterns) 1 and 3 has parallel communication, which goes from state 1 to state 2 and further to state 3. The presence of parallel paths in the graph contradicts the Law of Demeter. To eliminate this contradiction, parallel communication should be excluded from the scheme. To do this, in the matrix of transition probabilities ||πij, i = 1...10; j =1...10|| can be write π1.3=0. Communication from state 8 to state 6 is also excluded, based on similar reasoning: $\pi_{8\cdot6}$ = 0.

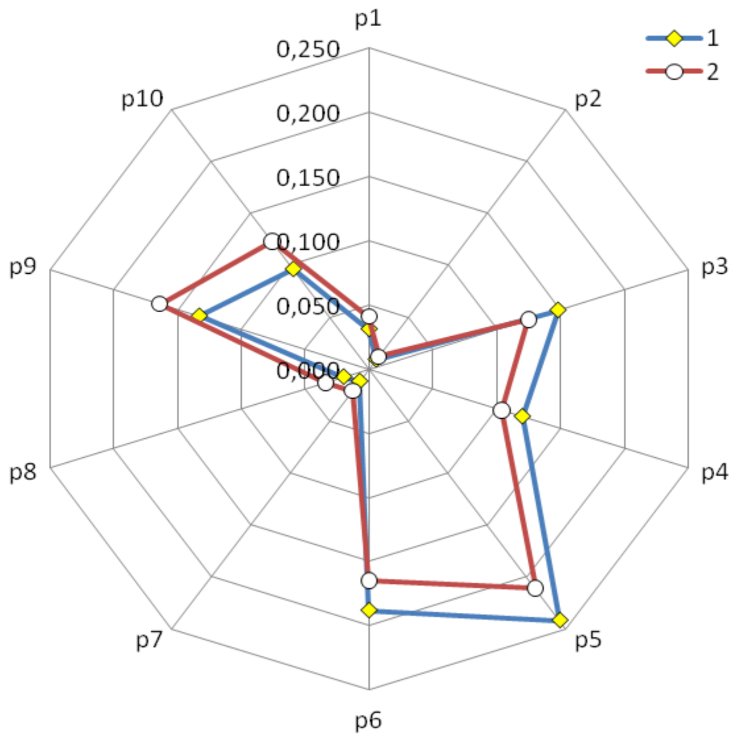


Figure 3: Change in the probabilities of software development process states for different schemes: 1 – basic scheme; 2 – modified scheme in accordance with the law of Demeter.

The results reflecting the project trajectory with the changed communication scheme are shown in Fig. 2b. As can be seen, the removal of parallel connections introduces some changes to the project trajectory. However, these changes are not disruptive – the trends of changing the project parameters practically coincide. Thus, at the 30th step, the highest probability of the state corresponds to process 5 – “Software support”, 9 – “Management in software engineering” and process 6 – “Software quality” (Fig. 2b).

A comparison of the values of transition probabilities for all states does not contradict the conclusion that both options, like fractals, give practically the same probabilities for all states (Fig. 3).

The advantages of the Law of Demeter are that systems designed in compliance with this law are simpler, and the created systems are more efficient in terms of support and introduction of new modifications. This is explained by the fact that design patterns (states, processes) are less dependent on the structure and connections with other processes.

The disadvantages of using the Law of Demeter in the context of building project management systems include the fact that it is sometimes quite difficult to determine the "near and far" states. Some processes have several inputs and outputs. There are no methods for formally determining the inconsistencies of the Law of Demeter.

5. Conclusions

The Law of Demeter significantly impacts software development and IT project management, improving the efficiency and adaptability of project structures. By introducing modularity and reducing dependencies between system components, this law simplifies project management, analysis, and modification throughout their lifecycle, contributing to the prospective development of project management.

Research has shown that project management schemes built on this law are more resilient to changing requirements. Their simplicity facilitates modifications and maintenance. Transforming cognitive models into Markov chains enables quantitative assessment and forecasting of project outcomes, providing a clearer understanding of their state dynamics over time.

Despite the numerous advantages of applying the Law of Demeter to managing complex systems, certain challenges are also identified. However, these complexities do not diminish the benefits of utilizing this law in IT projects. Overcoming these challenges can be a source of inspiration and motivation for project managers and IT professionals.

Further research is essential and should aim to refine methods for implementing the Law of Demeter in project management, particularly to overcome its limitations and expand its applicability to more complex project environments. Such development is a crucial direction that requires collective efforts for successful realization.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] ISO/IEC/IEEE 24765-2010 Systems and software engineering - Vocabulary.

[2] ISO/IEC TR 19759:2005, Software Engineering — Guide to the Software Engineering Body of Knowledge (SWEBOK).

[3] W. Stevens, G. Myers, L. Constantine, “Structured Design”, IBM Systems Journal, 13 (2), 115–139, 1974.

[4] Stelting, S. Using Java Templates. (2002) Professional Library, M.: Williams Publishing House.

[5] Demeter: Aspect-Oriented Software Development URL: http://www.ccs.neu.edu/research/demeter/.
[6] Taleb, Nassim Nicholas. (2009) The Black Swan. Under the Sign of Unpredictability, M.: Publishing house "KoLibri", 528.
[7] Yitzhak Adizes, 2020-2025 will go down as the most memorable disaster in history, 2021. URL: https://adizes.me/posts/itskhak-adizes-2020-2025-samaya-zapominayushchayasya- katastrofa.
[8] Who We Are. 2022. URL: https://www.vmedu.com/AboutUs/Who-We-Are.
[9] K. A. Mozhey, D. V. Lukianov, Aspects of the practical implementation of mass open online courses in the educational process of a higher educational institution. Actual problems of humanitarian education 2 (2017) 41–45.
[10] L. Chernova, S. Titov, S. Chernov & other, Development of a formal algorithm for the formulation of a dual linear optimization problem. Eastern-European Journal of Enterprise Technologies 4(100) (2019) 28-36.
[11] Robotic Process Automation (RPA) for Education Industry, 2021. URL: https://automationedge.com/rpa-for-education-industry/.
[12] K. Kolesnikova, T. Olekh, D. Lukianov, D. J. Obenewaa, Markov Principles of Project Effectiveness, in: IEEE International Conference on Smart Information Systems and Technologies SIST, Nur-Sultan, 2021, pp. 1-6. doi: 10.1109/SIST50301.2021.9465881.
[13] Kolesnikova K., Myrzakerimova A., Alpysbay N., Olekh T. Transforming Cognitive Maps into Markov Process Models for Software Development Projects (2023) SIST 2023 - 2023 IEEE International Conference on Smart Information Systems and Technologies, Proceedings, pp. 318 – 321 DOI: 10.1109/SIST58284.2023.10223535
[14] S. Bushuyev, D. Bushuiev, A. Zaprivoda, J. Babayev, Ç. Elmas, Emotional infection of management infrastructure projects based on the agile transformation. CEUR Workshop Proceedings 2565 (2022) 1–12.
[15] A. Bondar, S. Bushuyev, V. Bushuieva, S. Onyshchenko, Complementary strategic model for managing entropy of the organization. CEUR Workshop Proceedings 2851 (2021) 293–302.
[16] Freemium or Free Trial - which business model to choose when selling software?, 2021. URL: https://habr.com/ru/post/234579/.
[17] Free-to-play games: how to make them successful, 2021. URL: https://habr.com/ru/company/nevosoft/blog/137423/.
[18] D. Lukianov, K. Mazhei, V. Gogunskii. Transformation of the International Project Management Association Project Managers Individual Competencies Model, in: Proceedings of IEEE International Conference on Advanced Trends in Information Theory (ATIT-2019), Kyiv, Ukraine, 2019, pp. 506-512. doi: 10.1109/ATIT49449.2019.9030486.
[19] Neve Henry R., The Space of Dr. Deming: Principles for Building a Sustainable Business. Alpina Business Books, 2005.
[20] Decision making under uncertainty. Criteria for situation analysis, 2022. URL: http://it.kgsu.ru/IO/io_013.html.
[21] 7 mistakes in the development of an MVP product. URL: https://artjoker.ua/ru/blog/7-oshibok-razrabotki-mvp-produkta/.
[22] S. D. Bushuyev, Integrated computer-aided system for project management under market conditions. Journal of Computer and Systems Sciences International 32 (5) (1994) 132–138.
[23] V. Gogunskii et al., Lifelong learning is a new paradigm of personnel training in enterprises. Eastern European Journal of Advanced Technologies 4 (2) (2016) 4–10. doi: 10.15587/1729-4061.2016.74905.
[24] Kolesnikova K., Mezentseva O., Mukatayev T. Analysis of Bitcoin Transactions to Detect Illegal Transactions Using Convolutional Neural Networks (2021) SIST 2021 - 2021 IEEE International Conference on Smart Information Systems and Technologies, art. no. 9465983 DOI: 10.1109/SIST50301.2021.9465983

[25] R. Grant Continuing education-does it make for a more competent practitioner? Australian Journal of Physiotherapy 40 (1996) 33–37. doi: 10.1016/S0004-9514(14)60621-8.
[26] Careers and learning, 2021. URL: https://www2.deloitte.com/con-tent/dam/Deloitte/ec/Documents/deloitte-analytics/Estudios/Capi2.pdf.
[27] The European skills passport, 2022. URL: http://yourcompetences.com/en/toolbox-uk/pendant-la-realisation-2/passeport-europeen-competences/.
[28] Grabis J., Haidabrus B., Druzhinin E., Kolesnikova K. Deployment and Release Management Process in Agile Digital Projects (2024) Lecture Notes in Mechanical Engineering, pp. 124 - 136 DOI: 10.1007/978-3-031-61797-3_11
[29] Education: from the closure of educational institutions to the resumption of their work, 2022. URL: https://ru.unesco.org/covid19/educationresponse.
[30] Planning for a Safe Campus Reopening, 2021. URL: https://www.harvard.edu/coronavirus/planning-2020-21.
[31] HarvardX. Free online courses from Harvard University, 2021. URL: https://www.edx.org/school/harvardx.
[32] The Latest in Translation Devices, 2022. URL: https://www.nytimes.com/2019/11/07/travel/the-latest-in-translation-devices.html.