УДК 004.8

МАТЕМАТИЧНА МОДЕЛЬ АНАЛІЗУ ФАКТОРІВ ВПЛИВУ ПІДТРИМКИ ПРОГРАМНИХ КОМПЛЕКСІВ

А. І. Пукач, В. М. Теслюк

Інститут комп'ютерних наук та інформаційних технологій,
Національний університет «Львівська Політехніка»,
вул. С. Бандери, 28а, Львів, 79000, Україна

Розроблено математичну модель аналізу факторів впливу підтримки програмних комплексів. Модель дає змогу здійснити розрахунок ймовірностей приналежності нейронів прихованих шарів відповідних побудованих моделей багатошарового перцептрона, що інтерпретують процеси суб'єктивного сприйняття об'єкта (програмного комплексу чи його підтримки) до факторів впливу, що впливають на результати суб'єктивного сприйняття цього об'єкта відповідними суб'єктами взаємодії з ним. Результати, отримані на виході розробленої математичної моделі, є важливими для подальшого дослідження проблематики автоматизації підтримки програмних комплексів.

Ключові слова: *підтримка, програмний комплекс, фактори впливу, модель.*

Постановка проблеми. Основною досліджуваною науково-прикладною проблемою є втрата границь факторів впливу в результаті використання штучних нейронних мереж типу багатошарового перцептрона (БП) в моделях підтримки програмних комплексів, що зумовлено самою природою багатошарового перцептрона, відповідно до якої нейрони прихованих шарів БП не мають жодного функціонально-смислового навантаження, а слугують лише для підтримки чи покращення функціонування самого БП.

Мета статті — представлення розробленої математичної моделі, що дає змогу здійснити аналіз та відновити розподіл факторів впливу, який втрачений у результаті використання штучних нейронних мереж типу багатошарового перцептрона в моделях підтримки програмних комплексів.

Аналіз останніх досліджень та публікацій. Автоматизація уже проникла в значну кількість напрямів навколо програмних комплексів (чи програмного забезпечення), серед яких можна виокремити такі найбільш вагомі, базові та ключові напрями:

1. Автоматизація тестування програмного забезпечення. Можливо, один з перших напрямів автоматизації. Тому на сьогодні, а також згідно з інформацією поданою у праці [12], йдеться про автоматизацію тестування на основі технологій та підходів штучного інтелекту.

2. DevOps автоматизація. Однією з базових фундаментальних праць в цьому напрямі є праця [3]. У публікації [15] наведена цілісна методологія управління

знаннями DevOps. Відповідно до даних, наведених у праці [10], 80 % практиків програмного забезпечення, які брали участь у дослідженні, повідомили, що збірки програмного забезпечення найпростіше автоматизувати, за ними йдуть пакування та розгортання програмного забезпечення (51,2 % і 43,9 %, відповідно). У праці [11] підкреслено важливість впровадження DevOps в життєвий цикл розробки кожного програмного комплексу, а також той факт, що DevOps передбачає не лише зміни в процесах, а й зміну методу самої розробки програмного комплексу. У публікації [9] наведена автоматизація CI/CD (комбінована практика безперервної інтеграції та безперервної доставки) як однієї з найбільш істотних складових сучасної методології DevOps.

3. Автоматизація прийняття рішень, у праці [13] представлена структура проактивної підтримки прийняття рішень під назвою RADAR, заснована на дослідженнях автоматизованого планування в штучному інтелекті. Мабуть, найбільшої популярності підсистеми автоматизованого прийняття рішень здобули у використанні смарт-систем, таких як, наприклад, «розумний дім», що наведено у праці [4]. Проте активно цей напрям розвивається також і в сфері розробки програмного забезпечення, зокрема наведених у працях [8], де представлена автоматична класифікація нефункціональних вимог програмних продуктів на основі відгуків реальних користувачів, а також [5], де представлена автоматична класифікація функціональних і нефункціональних вимог за допомогою керованого машинного навчання. Також у публікації [6] подане порівняння прийняття рішень на основі даних (DDDM) до автоматизованого прийняття рішень на основі моделі на основі даних (MBDM). Фактично на сьогодні автоматизовані системи прийняття рішень вже давно покинули межі як смарт-систем, так і розробки програмного забезпечення і вийшли на масмаркет, зокрема у праці [2] Європейська організація споживачів чітко аналізує збільшення використання автоматизованого прийняття рішень на основі алгоритмів для комерційних транзакцій та його вплив на функціональність споживчих ринків і суспільств. Водночас у працях [7] та [1] піднімаються питання проблематики стрімкого росту впливу автоматизованих систем прийняття рішень та спричинених ними викликів та ризиків. Та навіть попри ці ризики, інтерес до цих систем продовжує зростати і вони продовжують розвиватись.

Проте ми пропонуємо дещо більш глобальний підхід та концепцію щодо дослідження процесів автоматизації підтримки програмних комплексів, а саме: ми підіймаємося на наступний (вищий) рівень, що пов'язує усі описані прояви автоматизації — від тестування до прийняття рішень. І на цьому наступному вищому рівні нас цікавить проблема впливу різноманітних факторів впливу на результати представлення об'єкта чи процесу підтримки програмних комплексів усіма учасниками (суб'єктами взаємодії) цієї підтримки — від тестувальника до менеджера з продажів підтримуваного програмного комплексу та реальних користувачів. Адже кожен із суб'єктів, що прямо чи опосередковано взаємодіє з підтримуваним програмним комплексом, свідомо чи підсвідомо стикається із множиною як універсальних, так й індивідуальних факторів впливу, що так чи інакше впливають на їх суб'єктивне результуюче сприйняття цього об'єкта підтримки.

Саме тому в контексті цієї праці нас цікавлять саме ці фактори впливу, а конкретніше — розроблення відповідної математичної моделі для дослідження цих факторів впливу, що надасть нам можливість їх подальшого дослідження, аналізу, диференціації, розрахунку, оцінки, визначення степеня та границь їх впливів тощо.

Моделі багатошарового перцептрона для інтерпретації суб'єктивного сприйняття об'єкта підтримки програмних комплексів.

На сьогодні існує багато різновидів штучних нейронних мереж. Усі вони цілком задовольняють нас з погляду представленої концепції. Проте в цій статті як приклад представлене використання саме моделі штучних нейронних мереж — багатошарового перцептрона (БП), описаного зокрема у праці [14]. Відповідно до запропонованої концепції факторів впливу, побудова моделей багатошарового перцептрона для інтерпретації суб'єктивного сприйняття об'єкта підтримки програмних комплексів здійснюється згідно з такими правилами: — вхідний шар нейронів — повинен відображати вхідні характеристики об'єкта дослідження (підтримки програмного комплексу); — нейрони прихованих шарів — повинні відображати фактори впливу на результати сприйняття суб'єктом взаємодії об'єкта дослідження; — вихідний шар нейронів з результатами роботи БП — повинен відображати вихідні результати сприйняття об'єкта дослідження суб'єктом взаємодії.

Вхідні дані для розробленої математичної моделі.

Основні вхідні дані для розробленої математичної моделі наведені нижче.

1. Розроблена та навчена на загальній вибірці відповідна модель БП для інтерпретації суб'єктивного сприйняття об'єкта підтримки програмних комплексів.

* важливо також зазначити, що дані для навчання та тестування моделі БП мають бути наведені у деперсоналізованій та нормалізованій формі.

2. Результати тестування БП на загальній (global) вибірці даних, а також результати тестування БН на окремих індивідуальних (local) вибірках даних для ізольованого впливу кожного окремо взятого фактора впливу (Dimension Factor).

Результати тестування мають містити:

- ідентифікатор коректності отриманого результату порівняно з очікуваним — оскільки нас цікавлять лише коректні результати роботи моделі БП, де актуальний результат збігається з очікуваним, в той час як некоректні результати ми просто відсіюємо і не враховуємо як вхідні дані для нашої математичної моделі (це важливий критерій коректності функціонування розробленої математичної моделі: некоректні результати тестування БП в жодному разі не мають потрапити в нашу математичну модель, інакше це призведе до отримання некоректних результуючих даних на виході нашої математичної моделі);
- сформовані зворотні ланцюжки максимальних ваг — від коректного активного вихідного нейрона БП — назад через усі приховані шари, аж до відповідного вхідного нейрона, промарковані відповідними факторами впливу.

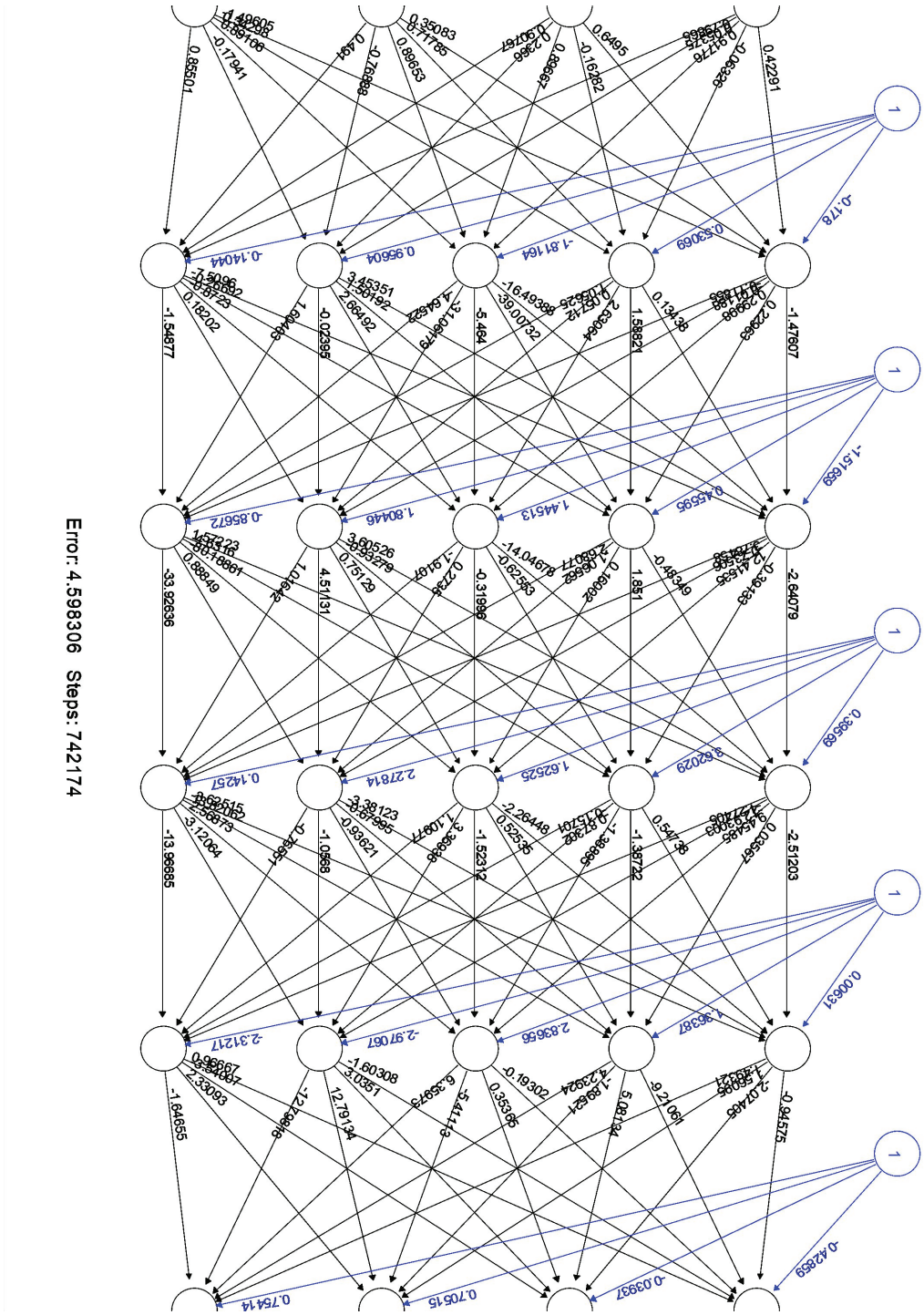

Рис. 1. Приклад розробленої моделі багатошарового перцептрона для інтерпретації суб'єктивного сприйняття об'єкта підтримки програмних комплексів

3. Серед усіх отриманих на попередньому кроці зворотних ланцюжків максимальних ваг необхідно вибрати лише унікальні (Unique Reverse Chain). Крім того, оскільки вхідні нейрони (як і вихідні нейрони) в отриманих зворотних ланцюжках максимальних ваг не становлять для нас інтересу з погляду їх приналежності до факторів впливу, нас цікавить лише приналежність до цих факторів впливу саме нейронів прихованих шарів, то ми також відкидаємо із отриманих унікальних зворотних ланцюжків вихідні та вхідні нейрони, залишаючи лише нейрони прихованих шарів.

Розроблення математичної моделі аналізу факторів впливу підтримки програмних комплексів. Відповідно до розробленої математичної моделі аналізу факторів впливу, розрахунок ймовірностей відбувається у кілька кроків.

На першому кроці розраховується процентне співвідношення частоти появи кожного із унікальних зворотних ланцюжків (в межах своїх факторів впливу) до загальної кількості усіх випадків (в межах своїх факторів впливу).

Нижче наведена формула для виконання розрахунків на першому кроці:

$$FoA\_local[i][j] = count(URC[j] \text{ in } F[i]) / count(URC[all] \text{ in } F[i]), \quad (1.1)$$

де - *FoA_local[i][j]* — частота появи (*Frequency of Appearance*) [j]-ого унікального зворотного зв'язку в межах [i]-ого фактору впливу;

- *count(URC[j] in F[i])* — кількість проявів [j]-ого унікального зворотного зв'язку (*Unique Reverse Chain*) в межах [i]-ого фактору впливу;

- *count(URC[all] in F[i])* — загальна кількість проявів усіх унікальних зворотних зв'язків, що існують в межах [i]-ого фактору впливу.

На другому кроці розраховується процентне співвідношення частоти появи кожного із унікальних зворотних ланцюжків (поза межами своїх факторів впливу, тобто на загальній вибірці випадків) до загальної кількості усіх випадків (цієї загальної вибірки випадків).

Нижче наведена формула для виконання розрахунків на другому кроці:

$$FoA\_global[i][j] = FoA\_local[i][j] * FI[i], \quad (1.2)$$

де - *FoA_global[i][j]* — частота появи [j]-ого унікального зворотного зв'язку [i]-ого фактору впливу в межах глобальної вибірки даних;

- *FI[i]* — індекс [i]-ого фактора впливу (Factor Index).

Індекс фактора впливу розраховується за такою формулою:

$$FI[i] = count(F[i] \text{ in General}) / count(\text{all in General}), \quad (1.3)$$

де - *count(F[i] in General)* — кількість проявів [i]-ого фактора впливу в межах глобальної вибірки даних;

- *count(all in General)* — загальна чисельність усіх кейсів глобальної вибірки.

На третьому кроці здійснюється розрахунок власне самих значень приналежності усіх спірних нейронів до їх відповідних конкурентних факторів впливу за такою формулою:

$$PoDNbtCDF[i][k] = Sum(FoA\_global[i][j](Neuron[k] \in URC[j])) / \\ / Sum(FoA\_global[i][j](Neurons[k] \in URC[all])), \quad (1.4)$$

де - *PoDNbtCDF[i][k]* — (*Probability of Disputed Neuron belongs to Competing Dimension Factor*) ймовірність приналежності спірного нейрона [k] до конкуруючого фактору впливу [I];

- *FoA_global[i][j](Neuron[k] Є URC[j])* — частота появи [j]-ого унікального зворотного зв'язку [i]-ого фактору впливу в межах глобальної вибірки даних (для тих унікальних зворотних зв'язків URC[j], до яких входить спірний нейрон Neuron[k]);

- *FoA_global[i][j](Neurons[k] Є URC[all])* — частота появи [j]-ого унікального зворотного зв'язку [i]-ого фактору впливу в межах глобальної вибірки даних (для усіх унікальних зворотних зв'язків усіх факторів впливу URC[all], до яких входить спірний нейрон Neuron[k]).

Тобто в глобальному розумінні суть концепції розробленої та наведеної математичної моделі полягає в тому, що ми:

– спочатку оцінюємо ймовірність появи того чи іншого унікального зворотного ланцюжка (з його прив'язкою до фактору впливу) в глобальній вибірці даних;
– а потім оцінюємо ймовірність появи кожного спірного нейрона у кожному з унікальних зворотних ланцюжків, до яких він належить, з прив'язкою кожного з цих ланцюжків до конкретного фактору впливу, забезпечуючи таким чином зв'язок між цим конкретним спірним нейроном та цим конкретним конкурентним фактором впливу.

Або, інакше кажучи (можливо, для ще кращого розуміння): у нас є перелік нейронів (усіх, зокрема і спірних) та перелік унікальних зворотних ланцюжків, до складу яких входять ці спірні нейрони, а самі ці унікальні зворотні ланцюжки також входять до складу факторів впливу (усіх, зокрема і конкуруючих).

Тож ми могли б просто порахувати кількість унікальних зворотних ланцюжків (до складу яких входить, припустимо, наш поточний досліджуваний спірний нейрон) одного єдиного конкретного фактору впливу (до якого належать ці унікальні зворотні ланцюжки) та розділити її на загальну кількість усіх унікальних зворотних ланцюжків (до складу яких входить наш поточний спірний нейрон) — безвідносно прив'язки цих унікальних зворотних ланцюжків до конкретного фактору впливу (а натомість просто беручи усі унікальні зворотні ланцюжки, до яких входить наш спірний нейрон, для усіх факторів впливу).

Проте такий розрахунок буде коректним лише тоді, коли:

– ймовірність появи у вибірці даних кожного фактору впливу у глобальній вибірці даних є рівною для усіх факторів впливу;
– ймовірність появи кожного унікального зворотного ланцюжка у глобальній вибірці даних є рівною для усіх унікальних зворотних ланцюжків.

Але в реальності це не так, і ми маємо конкретні числові значення як ймовірностей появи кожного із факторів впливу у глобальній вибірці даних, так і конкретні числові значення ймовірностей появи кожного із унікальних зворотних ланцюжків у цій же глобальній вибірці даних.

Саме тому при розрахунку співвідношення:

– кількості унікальних зворотних ланцюжків (до складу яких входить поточний спірний нейрон) одного єдиного конкретного фактору впливу (до якого належать ці унікальні зворотні ланцюжки);

– до загальної кількості усіх унікальних зворотних ланцюжків (до складу яких входить наш поточний спірний нейрон) для усіх факторів впливу.

Ми вже враховуємо попередньо прораховані числові значення як ймовірностей появи кожного із факторів впливу у глобальній вибірці даних, так і ймовірностей появи кожного із унікальних зворотних ланцюжків у цій же глобальній вибірці даних.

Ось що насправді означають усі ці розроблені та наведені формули математичної моделі аналізу факторів впливу.

На виході розробленої математичної моделі ми отримаємо конкретні числові значення ймовірностей приналежності нейронів прихованих шарів моделі багатошарового перцептрона до відповідних факторів впливу. Приклад реальних отриманих результатів наведений у табл. 1.

Таблиця 1

Приклад результатів, отриманих на виході розробленої математично моделі

	Ймовірність: [0]..-..[1]			
Спірний нейрон	1	2	3	4
HLN[3][1]	0.272	0.7252	-	0.00292
HLN[3][3]	-	-	0.7	0.3
HLN[2][0]	0.1726	-	0.8274	-
HLN[2][1]	0.00316	0.99685	-	-
HLN[2][4]	0.0176	0.97845	-	0.00394
HLN[2][2]	-	-	0.673	0.327
HLN[1][1]	0.00852	0.9915	-	-
HLN[1][4]	-	-	0.67	0.33
HLN[0][3]	0.002	0.28766	0.6974	0.013
HLN[0][1]	0.0067	0.72	0.165	0.1086
НЕ спірні нейрони				
HLN[3][0]	0	1	0	0
HLN[3][2]	0	0	0	1
HLN[1][0]	0	0	1	0
HLN[1][3]	1	0	0	0
HLN[0][2]	1	0	0	0

Висновки. Наведено розроблену математичну модель аналізу факторів впливу автоматизації підтримки програмних комплексів. Розроблена математична модель є унікальною та не має відомих аналогів. Основна науково-прикладна задача, яку вирішує розроблена математична модель, — відновлення границь факторів впливу при введенні моделей багатошарового перцептрона у моделі підтримки програмних

комплексів, що призводить до розмиття границь факторів впливу, а розроблена математична модель, власне, дає змогу певною мірою відновити ці втрачені границі факторів впливу. Результати, отримані за допомогою розробленої математичної моделі, дають можливість досліджувати подальші науково-прикладні задачі більш комплексної та глобальної науково-прикладної проблеми автоматизації підтримки програмних комплексів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Araujo T., Helberger N., Kruikemeier S., de Vreese C. H. In AI we trust? Perceptions about automated decision-making by artificial intelligence. *AI & SOCIETY*. 2020. 35 (3). 611–623. Doi: https://doi.org/10.1007/s00146-019-00931-w.
2. BEUC, 2018: Automated Decision Making And Artificial Intelligence - A Consumer Perspective. Report, 20 June 2018, Brussels. URL: https://www.beuc.eu/publications/beuc-x- 2018-058_automated_decision_making_and_artificial_intelligence.pdf.
3. Humble Jez, Farley David. Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation (3rd ed.). Addison Wesley, 2010. 463 p.
4. Jaihar J., Lingayat N., Vijaybhai P. S., Venkatesh G., Upla K. Smart home automation using machine learning algorithms, in: 2020 International Conference for Emerging Technology (INCET), IEEE, 2020. Pp. 1–4. doi: 10.1109/INCET49848.2020.9154007.
5. Kurtanović Z., Maalej W. Automatically classifying functional and non-functional requirements using supervised machine learning. In 2017 IEEE 25th International Requirements Engineering Conference (RE), 2017 Sep 4. Pp. 490−495.
6. Landau I., Landau V. From data driven decision making (DDDM) to automated datadriven model based decision making (MBDM). 2017. hal-01527766. URL: https://hal.archives-ouvertes.fr/hal-01527766.
7. Larus J., Hankin C., Carson S. G., Christen M., Crafa S., Grau O., … Werthner H. When Computers Decide: European Recommendations on Machine-Learned Automated Decision Making. New York, NY, USA: ACM, 2018. Doi: https://doi.org/10.1145/3185595.
8. Lu M., Liang P. Automatic classification of non-functional requirements from augmented app user reviews. In Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, 2017 Jun 15. Pp. 344−353.
9. Mohanty L., Tajammul M. DevOps CI Automation Continuous Integration. *International Research Journal of Engineering and Technology (IRJET)*. 2022. Vol. 9. Issue 3. Pp. 1221–1226.
10. Salameh H. The impact of devops automation, controls, and visibility practices on software continuous deployment and delivery, in: Proceedings of the 2nd International Conference on Research in Management and Economics, 2019. Pp. 22–46.
11. Sushma T. N. Automation of software development using devops and its benefits. *IJERT*. 2020. 9 (6). IJERTV9IS060369. DOI: 10.17577/IJERTV9IS060369.
12. Trudova A., Dolezel M., Buchalcevova A. Artificial intelligence in software test automation: A systematic literature review, in Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering, 2020. Pp. 181–192. doi: 10.5220/0009417801810192.

13. Vadlamudi Satya Gautam et al. Proactive Decision Support using Automated Planning, arXiv preprint arXiv:1606.07841. 2016.
14. Vang-Mata R. Multilayer Perceptrons: Theory and Applications, New York, Nova Science Publishers, 2020. 143 p.
15. Wettinger J., Andrikopoulos V., Leymann F. Automated capturing and systematic usage of devops knowledge for cloud applications, in Cloud Engineering (IC2E), 2015 IEEE International Conference on, 2015. Pp. 60–65.

REFERENCES

1. Araujo, T., Helberger, N., Kruikemeier, S., & de Vreese, C. H. (2020). In AI we trust? Perceptions about automated decision-making by artificial intelligence: AI & SOCIETY, 35 (3), 611–623. Doi: https://doi.org/10.1007/s00146-019-00931-w (in English).
2. BEUC, 2018: Automated Decision Making And Artificial Intelligence - A Consumer Perspective. Report, 20 June 2018, Brussels. Retrieved from https://www.beuc.eu/publications/beuc-x-2018-058_automated_decision_making_and_artificial_intelligence.pdf (in English).
3. Humble, Jez, & Farley, David. (2010). Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation (3rd ed.). Addison Wesley (in English).
4. Jaihar, J., Lingayat, N., Vijaybhai, P. S., Venkatesh, G., & Upla, K. (2020). Smart home automation using machine learning algorithms, in: 2020 International Conference for Emerging Technology (INCET), IEEE, 1–4. doi: 10.1109/INCET49848.2020.9154007 (in English).
5. Kurtanović, Z., & Maalej, W. (2017). Automatically classifying functional and non-functional requirements using supervised machine learning. In 2017 IEEE 25th International Requirements Engineering Conference (RE), 490−495 (in English).
6. Landau, I., & Landau, V. (2017). From data driven decision making (DDDM) to automated datadriven model based decision making (MBDM). hal-01527766. Retrieved from https://hal.archives-ouvertes.fr/hal-01527766 (in English).
7. Larus, J., Hankin, C., Carson, S. G., Christen, M., Crafa, S., Grau, O., … & Werthner, H. (2018). When Computers Decide: European Recommendations on Machine-Learned Automated Decision Making. New York, NY, USA: ACM, Doi: https://doi.org/10.1145/3185595 (in English).
8. Lu, M., & Liang, P. (2017 Jun 15). Automatic classification of non-functional requirements from augmented app user reviews. In Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, 344−353 (in English).
9. Mohanty, L., & Tajammul, M. (2022). DevOps CI Automation Continuous Integration: International Research Journal of Engineering and Technology (IRJET), 9, 3, 1221–1226 (in English).
10. Salameh, H. (2019). The impact of devops automation, controls, and visibility practices on software continuous deployment and delivery, in: Proceedings of the 2nd International Conference on Research in Management and Economics, 22–46 (in English).
11. Sushma, T. N. Automation of software development using devops and its benefits. IJERT. 2020. 9 (6). IJERTV9IS060369. DOI: 10.17577/IJERTV9IS060369 (in English).
12. Trudova, A., Dolezel, M., & Buchalcevova, A. (2020). Artificial intelligence in software test automation: A systematic literature review, in Proceedings of the 15th International

Conference on Evaluation of Novel Approaches to Software Engineering, 181–192. doi: 10.5220/0009417801810192 (in English).
13. Vadlamudi, Satya Gautam et al. (2016). Proactive Decision Support using Automated Planning, arXiv preprint arXiv:1606.07841 (in English).
14. Vang-Mata, R. (2020). Multilayer Perceptrons: Theory and Applications, New York, Nova Science Publishers (in English).
15. Wettinger, J., Andrikopoulos, V., & Leymann, F. (2015). Automated capturing and systematic usage of devops knowledge for cloud applications, in Cloud Engineering (IC2E), 2015 IEEE International Conference on, 60–65 (in English).

doi: 10.32403/0554-4866-2024-1-87-75-85

MATHEMATICAL MODEL FOR ANALYSIS OF INFLUENCING FACTORS ON SOFTWARE COMPLEXES SUPPORT

A. I. Pukach, V. M. Teslyuk

ACS Department, Institute of Computer Sciences and Informational Technologies,
Lviv Polytechnic National University,
28a, S. Bandera St., Lviv, 79000, Ukraine
andriipukach@gmail.com,
vasyl.teslyk@gmail.com

A mathematical model for analysis of influencing factors of the software complexes support automation is developed and represented in this article. The model makes it possible to calculate the probability of the belonging of the multi-layer perceptron hidden layer neurons to the influencing factors of the subjective perception of the supported programming complex by the appropriate relevant subject(s) of interaction with it. The methodological base of the research consists of: methods of research of artificial neural networks, in particular multilayer perceptron models (used to interpret the subjective perception of the object of software complexes support), methods of mathematical modeling (used to develop a mathematical model of the analysis of factors influencing the support of software complexes), as well as methods of computer design, computer modeling and computer programming for modeling the developed mathematical model, and obtaining and analyzing the relevant results of its work. Among the main theoretical results obtained, a mathematical model is developed, which enables the analysis and calculation of probabilities of belonging of the multilayer perceptron models hidden layers neurons (which interpret the processes of subjective perception of an object) to the influencing factors that affects the results of subjective perception of the object by the subjects of interaction with this object. Among the main scientific and practical results – all results obtained at the output of the developed mathematical model are important for further research into the problems of software complexes support automation. The developed mathematical model is unique and has no known analogues. The main scientific

and applied problem solved by the developed mathematical model is the problem of restoring the boundaries of the influencing factors lost (blurred) when introducing multilayer perceptron models into the models of software complexes support. The main practical value is that the developed mathematical model, in fact, makes it possible to determine and establish (restore) these lost (blurred) boundaries of influencing factors. The results obtained with the help of the developed mathematical model provide an opportunity to research further scientific and applied issues of the more complex and global scientific and applied problem of software complexes support automation.

Keywords*: support, program complex, influencing factors, model.*

Стаття надійшла до редакції 21.03.2024.
Received 21.03.2024.