DOI: 10.18372/2310-5461.46.14806
УДК 004.415

С. С. Бучик д-р техн. наук, доц.
Київський національний університет імені Тараса Шевченко,
orcid.org/0000-0003-0892-3494
e-mail: buchyk@knu.ua;

О. О. Писарчук, д-р техн. наук, проф.
Національний авіаційний університет
orcid.org/0000-0001-5271-0248
e-mail: pusarchuk@nau.edu.ua;

В. В. П'янкова
Національний авіаційний університет
orcid.org/0000-0002-9017-2396
e-mail: tnstp35@gmail.com.

ЕВОЛЮЦІЯ РОЗВИТКУ ЕМПІРИЧНОЇ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ (ОГЛЯД)

Вступ

Еволюція програмного забезпечення (ПЗ) означає динамічну поведінку програмних систем, коли вони підтримуються і розвиваються протягом усього життя. Еволюція ПЗ особливо важлива, оскільки за рахунок саме можливості еволюціонувати системи стають все більш довговічними.

Складність емпіричної інженерії програмного забезпечення (ЕІПЗ) так як і проведення емпіричних досліджень в інших галузях знань пов'язана зі збором інформації про об'єкт дослідження, виконанні експерименту та висуненні гіпотез.

Емпірична інженерія програмного забезпечення використовується для відповіді на емпіричні питання відносно ПЗ, яке розробляється. При дослідженні формуються теорії, припущення та гіпотези. З цих гіпотез робляться прогнозування конкретних подій. Прогнозування перевіряються відповідними експериментами. Залежно від результатів експерименту, формулювання підтверджуються або спростовуються.

Емпіричні дослідження дають емпіричні дані, які потім можуть бути проаналізовані на статистичну значущість [1] для подальшого прийняття рішень.

Дослідження еволюції розвитку ЕІПЗ дає змогу показати інформацію у систематизованому та впорядкованому вигляді, це надасть змогу побачити, які питання ще не розглядалися, або що було розкрито не цілком, що і зумовлює *актуальність* тематики статті.

Аналіз останніх досліджень і публікацій

У цілому тематика, яка присвячена ЕІПЗ, висвітлена достатньо у працях зарубіжних та вітчизняних учених. Але саме тематиці еволюції розвитку ЕІПЗ в Україні не приділяється багато уваги. Одним з учених, що досліджували цю тематику був О. П. Дишлевий в науковій статті «Метод та засіб для емпіричних досліджень програмного забезпечення». Стаття присвячена розробленню методу та засобу для емпіричних досліджень ПЗ [2]. Також М. Ф. Радішевський, Ю. М. Рябокінь у своїй пріці «Оцінювання об'єктно-орієнтованих програмних систем на етапі проектування» досліджували застосування метрик (NSS, NKC, NSUB, DIT, NOC, CBO, RFC, WMC) та їх вплив на якісні характеристики програмного забезпечення [3].

Що стосується зарубіжних учених, то дослідженням еволюції ЕІПЗ займались М. М. Леман та Л. Беладі, Н. А. Саймон, А. Ньюел, Чонг Хок Юен та ін.

Так М. М. Леманом та Л. Беладі були сформульовані перші емпіричні закони еволюції ПЗ. Чонг Хок Юеном було дослідження п'ять законів динаміки еволюції Беладі і Лемана. Він

повторно аналізує три різних системи від Беладі і Лемана, плюс кілька інших систем, і дивиться на різні залежні змінні, включаючи кількість і відсоток модулів.

Тецуо Тамай і Йошук Торіміцу зробили висновки, щодо необхідності заміни ПЗ після 5 років використання. Кук і Рош знайшли докази законів еволюції, посилаючись на постійні зміни, зростаючу ентропію [4].

Н. А. Саймон та А. Ньюел ще в 1976 році розглядали обчислення як емпіричну науку, на що вказує сама назва їх спільної публікації «Комп'ютер. Наука як емпіричне дослідження», в якій предметом їх дослідження виступала інформатика, яка ними трактувалась як: «Інформатика — це вивчення явища навколишніх комп'ютерів». Методологія обчислення, проведена Саймоном і Ньюелом, заснована на поняттях спостереження та експеримент, але розуміння цього відрізняється від вузької концепції емпіризму. Поняття спостереження та експериментів тут базуються на широкій основі зрозумілого емпіризму [5].

Також авторами статті у праці [6] наведені методи підтримки прийняття рішень, які можуть бути застосовані для проведення емпіричних досліджень ПЗ. Кожен з науковців зробив великий вклад до розвитку емпіричних методів програмного забезпечення. Але не зважаючи на це, в розширеному вигляді еволюція розвитку ЕІПЗ не була представлена.

Таким чином ***метою*** статті є проведення дослідження щодо еволюції розвитку емпіричної інженерії програмного забезпечення.

Виклад основного матеріалу

Емпірична інженерія програмного забезпечення — сукупність дій для отримання знань з метою кращого розуміння аспектів розробки програмного забезпечення. Результатом дій є ряд тверджень щодо визначеного переліку проблем. Ці твердження являються відповідями на поставлені запитання та підтвердженням чи спростуванням гіпотез. Еволюційні явища в програмних областях не обмежуються програмами та пов'язаними артефактами, технічними характеристиками, проектами та документацією.

Застосування, визначення носія, цілі, парадигми, алгоритми, мови, практики використання, підпроцеси та процеси розробки ПЗ тощо — також є еволюційними явищами.

Ці еволюціонуючі суб'єкти взаємодіють і впливають один на одного. Якщо їх еволюція повинна бути дисциплінованою, то відповідна еволюція процесів повинна бути запланована та керована. Щоб вони були освоєними, вони повинні бути зрозумілими і освоєними індивідуально та спільно [6].

Визнання еволюції ПЗ, його ідентифікація як дисципліноване явище та його подальше вивчення було викликано доповіддю 1968–1969 рр., що має назву «Процес програмування» (Lehman 1969) [7].

Зокрема, в дослідженні розглянуті емпіричні дані про зростання операційної системи IBM OS/360-370. Було зроблено висновок, що еволюція системи вимірюється зростанням розміру над послідовними релізами, де відображається регулярність, яка навряд чи була в першу чергу рішенням людини [4].

М. М. Леман виділив такі типи програм: S — програми написані в строгій відповідності із специфікацією того, що програма може робити; P — програми, які реалізують процедури, що повністю визначають їх поведінку; E — програми, що здійснюють роботу в умовах реального світу, тобто істотно залежні від середовища свого функціонування, а тому потребують адаптації до тих або інших зовнішніх вимог [6].

Ще в 1970-х роках М. М. Лемман почав формулювати закони еволюції програмного забезпечення, усвідомивши необхідність розробки програмних систем [7]. При цьому виділив закони Е-типу (табл. 1).

За словами Баррі та Л. Беладі ці закони (табл. 1) можна розділити на три категорії:

I. Закони про еволюцію характеристик програмних систем;

II. Закони, що стосуються організаційно-економічних обмежень на розвиток ПЗ;

III. Мета-закони еволюції ПЗ.

Таблиця 1

Закони Е-Типу Software Evolution

Номер (рік)	Назва	Заява
I (1974 р.)	Продовження зміни	Система Е-типу повинна постійно адаптуватися інакше стає все менш задовільною у вживанні
II (1974 р.)	Зростання складності	Складність системи зростає якщо не буде зроблена робота для підтримки чи зменшення складності
III (1974 р.)	Саморегулювання	Еволюція глобальної системи Е-типу регулюється зворотнім зв'язком

Закінчення табл. 1

Номер (рік)	Назва	Заява
IV (1978 р.)	Збереження організаційної стабільності	Швидкість роботи організації, що розвиває Е-тип, як правило, постійна над експлуатаційним терміном роботи цієї системи або сегментами її життя
V (1991 р.)	Збереження початкової форми	Загалом, зростання приросту (темпи зростання, тенденція) систем Е-типу обмежується потребою підтримки початкової форми
VI (1991 р.)	Продовження зростання	Функціональні можливості систем Е-типу повинні постійно вдосконалюватися, щоб забезпечити задоволення користувачів за час життя системи
VII (1996 р.)	Зниження якості	При вдосконалені строго адаптованих і розвинутих систем, якість системи Е-типу спадає.
VIII (1971/96)	Зворотній зв'язок	Еволюційні процеси Е-типу — це багаторівневі, багатоагентні системи зворотного зв’язку

Перше систематичне вивчення законів Беладі і Лемана було виконано одним із студентів Лемана Чонг Хок Юеном. Його робота була представлена в серії з трьох емпіричних робіт, опубліковані в 1985, 1987 і 1988 роках [8–10]. Автор повідомляє про результат, де застосовується дефект виявлення та корекції, включаючи спостереження. Час, який було виділено для виправлення дефекту, не збільшувався, як можна було очікувати через зростання складності системи. Автор повторно аналізує три різних системи та кілька інших систем, і дивиться на різні залежні змінні такі як: кількість модулів, відсоток оброблених даних, тривалість фази .

Після перегляду даних з попередніх досліджень, характерні для OC/360 не обов’язково досліджувати інші системи; перші два закони були підтверджені, решту законів не було досліджено. Однак автор зазначає, що останні закони більше засновані на людських організаціях, що займаються процесом технічного обслуговування ніж властивості самого ПЗ. Юен продовжує свою експертизу даних B-систем проводився аналіз часових рядів на кількість помилок.

Автор зазначає, що дані які вивчалися на попередніх етапах, він називає їх «глобальними» даними обслуговування (спостерігаються на глобальному рівні), як правило, демонструють мало результатів. Ці результати раніше були інтерпретовані як інваріація та визначені середні їх значення, що призвели до еволюційної динаміки.

У третій праці (1988) Юен розглядає технічне обслуговування великої частини ПЗ на рівні «О», а також на глобальному рівні. Крім тестів автор також використовує Time Series Analysis — методи спектрального аналізу (корелограма, хі-квадрат, авторегресія), а також лінійну фільтрацію (рухомі середні методи). Автор повідомляє про різні результати, в тому числі про ті, де зовнішні фактори відіграють велику роль у визначенні амплітуди кожного піка і інтервал між послідовними вершинами [11].

Тецуо Тамай і Йошук Торіміцу використовували опитувальник — огляд японських організацій для вивчення бізнесу. Заміна ПЗ відбувається протягом 5 років. Вони подають ряд описових результатів: ПЗ меншого масштабу має тенденцію мати коротший термін служби, адміністративний тип програм, як правило, мають більш тривалий термін служби, ніж підтримка типових програм (наприклад, продажна підтримка, виробництво) [4].

Кук і Рош займалися пошуком даних при факторному аналізі програмних показників. Автори роблять висновок, що інформаційні показники, такі як методи Halstead і McCabe, а також прості методи аналізу достовірніші. Вони також розглядали еволюцію системи протягом 18 місяців, і стверджували, що знайшли докази законів еволюції, посилаючись на постійні зміни, зростаючу ентропію [12].

Гефен і Шнабергер досліджували дві розподільні моделі модифікацій технічного обслуговування, щоб визначити, чи підтримує ПЗ однорідний розподіл. Автори вивчали звіти про проблеми ПЗ (SPR) з системи 4GL. Ці SPR характеризуються модифікаційним типом (коригувальні або адаптивні). Крім того, вони відстежують кількість модифікацій, викликаних попередніми. Автори відзначили, що рівень підтримки модифікації зменшуються в часі, але якщо розглядати в окремих фазах, які вони описують як «стабілізація», «вдосконалення», та «розширення», то навпаки збільшується [13].

Дишлевий О. П. у процесі дослідження визначив програмний код, який є придатним для повторного використання, за допомогою статистики та проведення розрахунків, на основі проведеного аналізу метрик 100 ПЗ. Пропонувалося автоматизувати процес оцінювання

шляхом визначення прямих метрик, які отримуються в результаті вимірювання програмного забезпечення, на непрямі метрики, які оцінюють експерти. Всі метрики (прямі та непрямі) були вибрані з розрахунку можливості їх використання для визначення повторно використовуваного програмного коду.

Запропоновано «Метод визначення залежностей між метриками програмного забезпечення за допомогою статистичного аналізу».

У загальному вигляді статистичний аналіз, який виконується з метою визначення залежностей, складається з трьох етапів: первинний статистичний аналіз, кореляційний аналіз та регресійний аналіз [3].

Цей метод характеризується наступним чином:

- відсутність визначення закону розподілу метрики;
- обов'язкове велике значення вибірки; використання розрахунку тільки парної рангової кореляції;
- відсутність перевірки точності коефіцієнтів кореляції;
- відсутність перевірки спільного закону розподілу метрик;
- побудова регресії методом лінеаризації.

М. Ф. Радішевський, Ю. М. Рябокінь досліджували різні метрики на етапі проектування для того, щоб вже на цьому етапі дізнатись складність системи. Складність системи є якісною характеристикою. Зменшення складності ПС дає змогу знизити трудомісткість проектування, розробки, тестування та супроводження, забезпечує простоту і надійність виробленої ПС. На етапі проектування оцінювання складності здійснюється оцінкою окремих компонентів системи та зв'язків між ними. Зв'язність та зчеплення класів мають аналогію зі зв'язністю та зчепленням модулів. Класи відповідають модулям, а функції — методам. Збільшення внутрішньої зв'язності та зменшення зовнішнього зчеплення знижує складність та сприяє забезпеченню надійності програмних систем [2].

Результати порівняльного аналізу еволюції розвитку емпіричної інженерії ПЗ наведено в табл. 2.

Таблиця 2

Емпіричне дослідження еволюції ПЗ

Автор	Публікація	Метод	Інформація	Залежні змінні	Статистична перевірка
Беладі і Леман (1976)	IBM системний журнал	Польове дослідження	21-випуск орієнтованих на користувачів	Номери послідовностей випуску, вік системи, розмір системи, кількість системних модулів	Багатовимірна регресія, автокореляція
Юен (1985)	IEEE Conference on Software Maintenance	Аналіз вторинних даних	5.000 «компонентів» протягом 19 місяців, 3000 KLOC	Посилання на джерело закладу пріоритету, опублікування, вплив машини, категорія виявлених помилок, час відгуку	Хі-квадрат, міра коефіцієнта непередбаченої ситуації, часові ряди, Т-статистика, авто- і крос-кореляції, розподіл Пуассона
Юен (1987)	IEEE Conference on Software Maintenance	Польове дослідження	Модулі від OS/360, OMEGA, EXECUTIVE, BD, B, DOS, систем CCSS	Обробка кумулятивних модулів, швидкість руху, частка оброблених модулів, розмір, інтервал випуску, чистий приріст	Запуск тесту. Тестування точок повороту. Тест довжини фази
Юен (1988)	IEEE Conference on Software Maintenance	Обстеження	Повідомлення, інформація, що видається комерційним користувачам системи з використанням того ж набору даних, як Юен (1985)	Релізи та кількість «повідомлень» на тиждень	Методи аналізу часових рядів, лінійна фільтрація

Закінчення табл. 2

Автор	Публікація	Метод	Інформація	Залежні змінні	Статистична перевірка
Тамаі і Торімітсу	IEEE Conference on Software Maintenance	Польове дослідження	95 систем від різних організацій, що звітують про роботу, виконану за попередні 5 років, Ri, програмне забезпечення Major Frame, 70 % COBOL.	Вік програмного ресурсу, розмір ПЗ до і після заміни, області застосування, фактори заміни	Вибіркова статистика, кореляція
Кук і Роєш	Журнал систем та програмного забезпечення	Польове дослідження	10 версій реального часу німецького телефонного комутаційного програмного забезпечення вийшло більше 18 місяців	Кількість функцій, кількість змінених функцій, кількість основних змін	Кореляції, експериментальний факторний аналіз з варімаксним обертанням
Гефен і Шенеберг	IEEE Conference on Software Maintenance	Польове дослідження	29 місяців звіти про програмні проблеми (SPRs), 250 KLOC	Тип модифікації (загальна кількість SPR, кількість коригуючих SPR, кількість адаптивних SPR), кількість нових додатків, кількість модифікацій, спричинених попередніми змінами	Лінійна регресія, «Вілкоксон-відповідні пари», «Тест звання», «Колмогоров-Смирнов», «Доброта придатного тесту»
Беладі	IEEE Conference on Software Engineering	Польове дослідження	25 програмних релізів з 10 різних систем на NASA Goddard.	Зусилля та розмір завдань різних видів обслуговування	Непараметричний тест Манна-Уітні U: регресія ОЛС
Леман	Міжнародний симпозіум метрики ПЗ	Польове дослідження	21 програмних релізів фінансового пакету	Розмір системи в модулях та кількість модулів	Найменші квадрати та модель зворотної квадратичної регресії; середня абсолютна помилка
О.П. Дишлевий	Науковий журнал НАУ. «Технічні науки»	Дослідження огляду	Метод обробки даних емпіричних досліджень ПЗ на основі статистичного аналізу	Кількість ПЗ для повторного використання - 100, метрики для ПЗ, коефіцієнт значущості	Лінійна регресія, кореляція
М. Ф. Раді-шевський, Ю. М. Рябокінь	Науковий журнал НАУ. «Технічні науки»	Дослідження огляду	Оцінювання об’єктно-орієнтованих програмних систем на етапі проектування	Розмір системи в модулях та кількість модулів, підсистем, методів	Кореляції, експериментальний факторний аналіз

Основні результати

До основних результатів можна віднести поступове ознайомлення з розвитком емпіричних методів, відношення дослідників до робіт інших науковців та аналіз основних законів емпіричної інженерії програмного забезпечення та їх методів.

Висновки

Проведений аналіз розвитку емпіричної програмної інженерії, досліджено роботи інших дослідників, систематизовано та узагальнено дані дослідів у вигляді таблиць. Підводячи підсумки можна сказати, що для кожного застосування емпіричного методу потрібно визначитися з метою, тобто проблемою досліду, використовувати різні підходи до вирішення певної задачі та чим більше експериментів, дослідів та практик тим краще. Кожен метод повинен застосовуватись в своїй сфері. В емпіричній інженерії програмного забезпечення об'єктом досліду є програма, і мета дослідів виявити зношення, конкурентоспроможність, економічну справедливість та час актуальності даної системи.

ЛІТЕРАТУРА

1. **Емпіричні** методи програмної інженерії : електронний конспект лекцій, КПІ. URL: http://tc.kpi.ua/content/kurs/EMPI/EMPI_konspect.pdf (Дата звернення: 24.01.2020)

2. **Радішевський М. Ф.,** Рябокінь Ю. М. Оцінювання об'єктно-орієнтованих програмних систем на етапі проектування. *Наукоємні технології.* 2009. №2. С. 74–78.

3. **Сидоров М. О.,** Дишлевий О. П. Метод та засіб для емпіричних досліджень програмного забезпечення. Наукоємні технології. 2009. №2. С. 59–64. DOI: 10.18372/2310-5461.2.5299

4. **Chris F.** Kemerer, Sandra Slaughter. An Empirical Approach to Studying Software Evolution. 1999. URL: https://plg.uwaterloo. ca/~migod/846/papers/kemerer-tse.pdf (Дата звернення: 27.01.2020)

5. **Paweł Polak.** Computing as Empirical Science — Evolution of a concept. 2016. URL: https://www.researchgate.net/publication/315437614_Computing_as_Empirical_Science_Evolution_of_a_Concept (Дата звернення: 04.02.2020)

6. **Бучик С. С.,** Кондратенко С. О., Писарчук О. О. Системи підтримки прийняття рішень: конспект лекцій. Житомир : ЖВІРЕ, 2006. 168 с..

7. **Michael W. Godfrey,** Daniel M. German. On the Evolution of Lehman's Laws. 2014. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.1636 (Дата звернення: 07.02.2020)

8. **C.K.S. Chong Hok Yuen.** An empirical approach to the study of errors in large software under maintenance, 1985. URL: https://www.semanticscholar.org/paper/An-empirical-approach-to-the-study-of-errors-in-Yuen/5849d0743d414eb889d11c0fbcf64806b7724b43 (Дата звернення: 10.02.2020)

9. **C.K.S. Chong Hok Yuen** A statistical rationale for evolution dynamics concepts. Proceedings of the IEEE Conference on *Software Maintenance.* 1987. P. 156–164.

10. **C.K.S. Chong Hok Yuen** On analyzing maintenance process data at the global and detailed levels: a case study. 1988. URL: https://ieeexplore.ieee.org/document/10170 (Дата звернення: 10.02.2020)

11. **Stephen Cook,** He Ji and Rachel Harrison Software Evolution and Software Evolvability. 2006. URL: https://pdfs.semanticscholar.org/bef8/e5f62c96df626ec24e28a0e03c41f3815985.pdf (Дата звернення: 15.02.2020)

12. **C.R. Cook and A. Roesch**, Real-Time Software Metrics, J. Systems and Software. 1994. URL: https://www.sciencedirect.com/science/article/pii/0164121294900655 (Дата звернення: 15.02.2020)

13. **D. Gefen and S.L. Schneberger**, The Non-Homogeneous Maintenance Periods: A Case Study of Software Modifications, 1996. URL: https://www.sciencedirect.com/science/article/pii/S0065245801800176. (Дата звернення: 16.02.2020)

Бучик С. С., Писарчук О. О., П'янкова В. В.

ЕВОЛЮЦІЯ РОЗВИТКУ ЕМПІРИЧНОЇ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ (ОГЛЯД)

Кожен рік світові IT-компанії представляють нове сучасне програмне забезпечення, але які є докази його розвитку. Проблемою визначення ефективності та розрахунком кількісних показників розвитку програмного продукту займається емпірична інженерія програмного забезпечення. Для того щоб якісно визначити ефективність, потрібно прослідити еволюцію програмного забезпечення. Еволюція програмного забезпечення відноситься до динамічної поведінки програмних систем, оскільки вони підтримуються та вдосконалюються протягом всього життєвого циклу. З цієї задачі виникає основне питання даної роботи дослідити та проаналізувати інформацію про етапи еволюції розвитку емпіричної інженерії програмного забезпечення. У цій статті представлено еволюцію філософських та методологічних міркувань щодо емпіризму та подано інформацію в зрозумілому та систематизованому вигляді. Були розглянуті роботи закордонних та

вітчизняних вчених, проаналізовано їх діяльність та які висновки були ними зроблені. В роботах вітчизняних вчених здебільшого розглядались сучасні системи та методи які визначали складність системи, структурованість і з цього робились висновки про конкурентоспроможність та зношення програми. В роботах вітчизняних авторів було досліджено опис класичних систем та саме їх розвиток відповідно до використання емпіричних методів. У цьому дослідженні ми простежуємо найважливіші поточні події в історії впливу на емпіричну інженерію. На основі зібраної інформації було створено таблицю, де в хронологічному порядку представлені вчені, що внесли свій здобуток в розвиток емпіричної інженерії. В результаті дослідження визначилось, що емпіричні методи, які використовуються відповідно до сфери застосування, мають єдину мету визначити: виявити зношення, конкурентоспроможність, економічну справедливість та час актуальності даної системи.

Ключові слова: програма; емпіричність; технологія; система; еволюція; ефективність.

Buchyk S., Pysarchuk O., Pyankova V.

EVOLUTION OF EMPIRICAL SOFTWARE ENGINEERING

Every year, global IT companies present new modern software, but what is the evidence of its development. Empirical software engineering deals with the problem of determining the efficiency and calculation of quantitative indicators of software product development. In order to qualitatively determine the effectiveness, you need to follow the evolution of software. Software evolution refers to the dynamic behavior of software systems as they are maintained and improved throughout the life cycle. From this task arises the main question of this work to investigate and analyze information about the stages of evolution of empirical software engineering. This article presents the evolution of philosophical and methodological considerations on empiricism and presents information in a clear and systematic way. The works of foreign and domestic scientists were considered, their activity was analyzed and what conclusions they made. In the works of domestic scientists mostly considered modern systems and methods that determine the complexity of the system, structure and from this conclusion were drawn about the competitiveness and wear of the program. In the works of domestic authors, the description of classical systems and their development in accordance with the use of empirical methods were studied. In this study, we trace the most important current events in the history of the impact on empirical engineering. Based on the collected information, a table was created, which presents in chronological order the scientists who have made their contribution to the development of empirical engineering. As a result of the study, it was determined that the empirical methods used in accordance with the scope of application have a single purpose to determine: to identify depreciation, competitiveness, economic fairness and relevance of the system.

Keywords: program; empiricism; technology; system; evolution; efficiency.

Бучик С. С., Писарчук О.О., Пьянкова В. В.

ЭВОЛЮЦИЯ РАЗВИТИЯ ЭМПИРИЧЕСКОЙ ИНЖЕНЕРИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (ОБЗОР)

Каждый год мировые ИТ-компании представляют новое современное программное обеспечение, но какие есть доказательства его развития. Проблемой определения эффективности и расчетом количественных показателей эволюции программного продукта занимается эмпирическая инженерия программного обеспечения. Для того чтобы качественно определить эффективность, нужно проследить эволюцию программного обеспечения. Эволюция программного обеспечения относиться к динамическому поведению программных систем, поскольку они поддерживаются и совершенствуются на протяжении всего жизненного цикла. Из этой задачи возникает основной вопрос данной работы исследовать и проанализировать информацию об этапах эволюции развития эмпирической инженерии программного обеспечения. В этой статье представлены эволюцию философских и методологических соображений относительно эмпиризма и дана информация в понятном и систематизированном виде. Были рассмотрены работы зарубежных и отечественных ученых. Проанализированы их деятельность и какие выводы были ими сделаны. В работах отечественных ученых в основном рассматривались современные системы и методы, которые определяли сложность системы, структурированность и с этого делались выводы о конкурентоспособности и износе программы. В работах отечественных авторов было исследовано описание классических систем и именно их развитие в соответствии с использование эмпирических методов. В этом исследовании мы прослеживаем важнейшие текущие события в истории влияния на эмпирическую инженерию. На основе собранной информации было создано таблицу, где в хронологическом порядке представлены ученые, внесшие свой вклад в развитие эмпирической инженерии. В результате исследования определилось, что эмпирические методы, которые используются в соответствии со сферой применения, имеют единственную цель определить: износ, конкурентоспособность, экономическую справедливость и актуальность данной системы.

Ключевые слова: программа; эмпиричность; технология; система; эволюция; эффективность.

Стаття надійшла до редакції 29.03.2020 р.
Прийнято до друку 08.06.2020 р.