DOI: https://doi.org/10.36910/6775-2524-0560-2021-44-23
УДК 00.004
Снігур Олена Миколаївна, к.пед.н., доцент
https://orcid.org/0000-0003-3515-9372
Національний педагогічний університет ім. М.П. Драгоманова

ІНСТРУМЕНТИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ ПРИЗНАЧЕНІ ДЛЯ ЗАБЕЗПЕЧЕННЯ ПРОЦЕСІВ ЖИТТЄВОГО ЦИКЛУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Снігур О. М. Інструменти програмної інженерії призначені для забезпечення процесів життєвого циклу програмного забезпечення. У статті розкрито інструменти програмної інженерії призначені для забезпечення процесів життєвого циклу програмного забезпечення. Визначено етапи еволюції методів та методологій застосовуваних для розробки та підтримки процесів життєвого циклу програмного забезпечення. Схематично представлено еволюцію методів та методологій застосовуваних для розробки та підтримки процесів життєвого циклу програмного забезпечення. Виділено чотири етапи еволюційного циклу: модель водоспаду (послідовна), модель фонтану (зворотна), ітеративна еволюційна модель, швидка комплексна модель розробки програмного забезпечення. Наголошено, що у зв'язку зі стрімким розвитком ІТ сфери практично для кожної фази життєвого циклу розробки програмного забезпечення були розроблені інструменти програмної інженерії. Для багатьох етапів існує велика кількість інструментів, які виконують ті ж або подібні функції. Деякі інструменти надають засоби, які охоплюють багато різних етапів, інші зосереджені на певному виді завдання, технології, мові чи проблемі розробки програмного забезпечення. Запропоновано граф функціональної приналежності кожного виду інструментів програмної інженерії до певного процесу життєвого циклу програмного забезпечення. Що дозволило візуально відстежити пристосованість кожного окремого виду інструментів до часового проміжку певного етапу. Описано набір інструментів і методів програмної інженерії для проектування програмного забезпечення, що допомагає забезпечити високу якість програм, відсутність помилок і простоту в обслуговуванні програмних продуктів та зазначається, що окреслена низка інструментів застосовується до аналізу, проектування та інженерних інструментів, але іноді використовується для позначення всіх інтегрованих програмних засобів, розгорнутих у проекті. Наголошено, що незважаючи на інтегральність, структурованість та універсальність багатьох інструментів програмної інженерії та масштабність послуг, які надаються ними, не завжди вони однаково застосовуються продавцями та їх дослідниками, за рахунок наявності специфічних приналежностей.
Ключові слова:інструменти, програмна інженерія, життєвий цикл, процес, програмне забезпечення.

Снигур Е. Н. Инструменты программной инженерии предназначены для обеспечения процессов жизненного цикла программного обеспечения. В статье раскрыты инструменты программной инженерии предназначенные для обеспечения процессов жизненного цикла программного обеспечения. Определены этапы эволюции методов и методологий применяемых для разработки и поддержки процессов жизненного цикла программного обеспечения. Схематически представлена эволюция методов и методологий применяемых для разработки и поддержки процессов жизненного цикла программного обеспечения. Выделены четыре этапа эволюционного цикла: модель водопада (последовательная), модель фонтана (обратная), итеративная эволюционная модель, быстрая комплексная модель разработки программного обеспечения. Отмечено, что в связи со стремительным развитием ИТ сферы практически для каждой фазы жизненного цикла разработки программного обеспечения были разработаны инструменты программной инженерии. Для многих этапов существует большое количество инструментов, которые выполняют те же или подобные функции. Некоторые инструменты предоставляют средства, которые охватывают много разных этапов, другие сосредоточены на определенном виде задачи, технологии, языке или проблеме разработки программного обеспечения. Предложен граф функциональной принадлежности каждого вида инструментов программной инженерии к определенному процессу жизненного цикла программного обеспечения. Что позволило визуально отследить приспособленность каждого отдельного вида инструментов к временному промежутку определенного этапа. Описан набор инструментов и методов программной инженерии для проектирования программного обеспечения который помогает обеспечить высокое качество программ, отсутствие ошибок и простоту в обслуживании программных продуктов и отмечается, что очерчена ряд инструментов применяется до анализа, проектирования и инженерных инструментов, но иногда используется для обозначения всех интегрированных программных средств, развернутых в проекте. Отмечено, что несмотря на целостность, структурированность и универсальность многих инструментов программной инженерии и масштабность услуг, предоставляемых ими, не всегда они одинаково применяются продавцами и их исследователями, за счет наличия специфических принадлежностей.
Ключевые слова: инструменты, программная инженерия, жизненный цикл, процесс, программное обеспечение.

Snihur Olena. Software engineering tools are designed to support software lifecycle processes. The article discloses software engineering tools designed to support software life cycle processes. The stages of evolution of methods and methodologies used to develop and maintain software life cycle processes are identified. The evolution of methods and methodologies used to develop and maintain software life cycle processes is schematically presented. There are four stages of the evolutionary cycle: the model of the waterfall (sequential), the model of the fountain (reverse), the iterative evolutionary model, the fast complex model of software development. It is emphasized that due to the rapid development of the IT sector, software engineering tools have been developed for almost every phase of the software development life cycle. For many stages, there are a large number of tools that perform the same or similar functions. Some tools provide tools that cover many different stages, while others focus on a specific type of task, technology, language, or software development problem. A graph of the functional affiliation of each type of software engineering tools to a certain process of the software life cycle is proposed. Which allowed us to visually track the adaptation of each type of tool to the time period of a particular stage. Describes a set of software engineering tools and techniques for software design that helps ensure high quality software, no errors and ease of maintenance of software products and notes that the outlined set of tools is used for analysis, design and engineering tools, but sometimes used to denote all integrated software deployed in the project. It is emphasized that despite the integrity,

structure and versatility of many software engineering tools and the scale of the services they provide, they are not always used equally by vendors and their researchers, due to the availability of specific accessories.

Key words: tools, software engineering, life cycle, process, software.

Вступ та постановка проблеми. Стрімке зростання попиту на комп'ютерні програмні програми неухильно зростає у міру зростання кількості користувачів, проблемних областей та областей застосування сучасного програмного забезпечення. Продовжують з'являтися нові програми обчислювальних технологій, такі як Інтернет речей [1], електронна комерція, зберігання даних, мультимедійні системи, мобільні обчислення та комплексне програмне забезпечення, багато з яких дуже вимогливі з точки зору програмного забезпечення, його розмірної складової та складності виконання. Так само, коли обчислювальне обладнання та мережі стають все більш потужними, програмне забезпечення, має тенденцію зростати, щоб відповідати (або перевищувати) можливості бази реалізації. Таким чином, останнім часом, програмні системи продовжують ставати більш глобальними та складнішими – і розробникам програмного забезпечення потрібні вдосконалені процеси, інструменти та методи для управління цією складністю. Крім того, інноваційні технологічні процеси розробки програмного забезпечення, такі як ітераційна розробка, швидка розробка додатків та екстремальне програмування, висувають більші вимоги до розробників щодо організації їх діяльності та артефактів програмного забезпечення. Нові методи розробки, такі як компонентна інженерія систем, інтеграція корпоративних програм та орієнтоване на технологічні аспекти програмування, потребують додаткового моделювання та підтримки управління. Таким чином, потреба в хороших інструментах програмної інженерії для підтримки незліченної кількості заходів, що відбуваються під час розробки програмного забезпечення, є набагато більшою ніж здається. Протягом багатьох років програмні засоби були розроблені для підтримки частин усього життєвого циклу розробки. Інструменти використовуються для опису та спільного використання програмних процесів, планування та управління командною роботою, збору інформації про продуктивність процесу та, зрештою, для покращення процесів. Інструменти допомагають у виявленні, кодифікації та валідації вимог. Інструменти автоматизованих специфікацій та проектування підтримують аналіз та проектування програмного забезпечення, а також, у багатьох випадках, включають інженерні функції в обидва боки, включаючи генерацію коду та зворотну інженерію. Середовища програмування та генератори програм забезпечують конструктори інтерфейсу користувача, конструктори баз даних та повідомлень, компілятори, редактори та налагоджувачі на вихідному рівні. Інструменти тестування варіюються від інструментів профілювання та моніторингу продуктивності до генераторів планів тестування, оракулів тестування, автоматизованих тестувальників інтерфейсу користувача та офіційного уточнення та доведення теорем. Засоби управління конфігурацією та контролю версій допомагають керувати великими сховищами програмних артефактів. Багато інструментів, зокрема інструменти для проектування програмного забезпечення та управління конфігураціями, забезпечують різну ступінь управління змінами та їх відстеження. Підтримка групового програмного забезпечення, часто інтегрована у зазначені інструменти, що полегшує спілкування та координацію команди.

Аналіз останніх досліджень і публікацій. Сучасна наукова думка стрімко розвивається з розвитком ІТ технологій. З'являється все більше робіт, в яких описуються інструменти програмної інженерії, механізми та принципи їх застосування щодо розробки програмного забезпечення.

О. М. Величко, О. В. Грабовський та Т. Б. Гордієнко [2] навели результати порівняльного аналізу для встановлення основних складових життєвого циклу програмного забезпечення для засобів вимірювальної техніки. Для цих досліджень використано як базові міжнародні стандарти щодо інженерії програмного забезпечення, так і специфічні міжнародні та регіональні документи щодо програмного забезпечення для засобів вимірювальної техніки. У [3] розглянуто спектр моделей предметної області "Метатехнологія програмування", які використовуються при конструюванні сервісно-орієнтованих технологій програмування в семантичному веб-середовищі. Визначені базові терміни і поняття цієї предметної області. Наведено концептуальну модель технології програмування й концептуальну та онтологічну моделі метатехнології програмування. Т. О. Говорущенко, Р. А. Малярчук [4] виконали аналіз сучасних технологій проектування, методологій та середовищ розроблення програмного забезпечення (ПЗ), в результаті якого було визначено основні переваги, недоліки та використовуваність технологій/ методологій/ середовищ. При наявній великій кількості технологій/методологій/середовищ частка проблемних проектів залишається сталою, а процес розроблення ПЗ залишається недетермінованим. Автори провели аналіз процесу оцінювання та вибору технології/ методології/ середовища і показали, що на даний момент в галузі панує повний суб'єктивізм вибору. Науковцями доведено актуальність задачі підтримки процесу вибору технології/ методології/ середовища для ПЗ.

Щодо принципів тестування програмного забезпечення варто відзначити роботу [5]. Авторами розглядаються основні поняття в області тестування програмного забезпечення, критерії вибору тестів, оцінка відтестованості проекту. Значна увага приділяється модульному та інтеграційному тестуванню, інтеграційному тестуванню для об'єктно-орієнтованого програмування. Розглядаються питання автоматизації тестування тощо. О. В. Марковець та А. І. Синько [6] описали механізми формування якісної технічної документації до програмного забезпечення з урахуванням чинного законодавства.

Із зарубіжних авторів варто відзначити такі роботи як: Shah, Unnati & Jinwala, Devesh & Patel, Sankita [7], Kobyliński A. [8], Basri, Sufyan & Kama, Nazri & Sarkan, Haslina & Ismail, S.A. & Haneem, Faizura [9], Bansiya J., and Davis C. [10], Maznan, Roslinda & Wan Kadir, Wan Mohd Nasir & Kadir, Wan [11], Synko A., and Peleshchshyn A. [12], Basri, Sufyan & Kama, Nazri & Haneem, Faizura & Ismail, S.A. [13], Burnstein I. [14] та інші.

Проте, враховуючи описані наукові набутки, за темою, питання розкриття інструментів програмної інженерії призначених для забезпечення процесів життєвого циклу програмного забезпечення залишається відкритим та потребує детального опрацювання.

Постановка завдання. Розкрити інструменти програмної інженерії призначені для забезпечення процесів життєвого циклу програмного забезпечення.

Викладення основного матеріалу дослідження. Еволюція методів та методологій застосовуваних для розробки та підтримки процесів життєвого циклу програмного забезпечення, на протязі багатьох років відбувалась поступово, змінюючи низько ітераційні моделі та моделі зворотного зв'язку на середньо ітераційні та високо ітераційні. Ці високоітераційні моделі також мають тенденцію використовувати менш офіційні процедури переходу від етапу до фази, порівняно з військовими системами та системами, що відповідають за безпеку, які використовують ретельний огляд та реєстрацію для контролю процесів та підтримки підзвітності. На рисунку 1 наведено цю еволюційну тенденцію: старі моделі програмного забезпечення, орієнтовані на завершення або майже завершення попередніх фаз розробки, з обмеженим зворотним зв'язком від пізніх фаз до попередніх.

Таким чином, програмні засоби, що використовуються для підтримки такої розробки, не потребували зусиль щодо обробки зворотного зв'язку – інструменти можуть бути розроблені з особливою орієнтацією на фазі життєвого циклу, механізми інтеграції можуть бути спрощеними (переважно зосереджені на експорті даних до інструментів, що використовуються на пізніх фазах), а підтримка спільної роботи, хоча і проявляється в деяких інструментах, часто делегована окремим продуктам, не пов'язаним із розробкою програмного забезпечення. Інноваційні технологічні моделі роблять більший акцент на специфікації, конструкції та прототипи, які швидко розвиваються, а також на їх потенційний розвиток у багатьох швидких ітераціях. Окремі блоки цієї роботи часто виконуються невеликими, тісно пов'язаними, можливо розподіленими групами розробників з періодичною інтеграцією частин. Інструменти, що використовуються для підтримки таких процесів, повинні забезпечувати або набагато більший діапазон охоплення діяльності, що підтримується протягом життєвого циклу, або забезпечувати більш складні механізми інтеграції, що підтримують повний двоспрямований обмін інформацією для програмної інженерії. Зростання популярності розробки розподіленого програмного забезпечення, що проявляється аутсорсингом, асинхронною роботою, командами телекомунікації та віртуальним програмним забезпеченням, – також висуває значно більші вимоги до розробників, які обмінюються інформацією та працюють із автоматизованою підтримкою для спільної роботи. Щоб бути дійсно ефективними, такі особливості співпраці повинні проявлятися у використовуваних програмних засобах [14]. Поширення нових технологій та методів програмної інженерії, включаючи технології проміжного програмного забезпечення та розробку на основі компонентів, та значно більших систем, які зазвичай розробляються, збільшує вимоги до програмних засобів. Розробникам потрібна покращена підтримка на високому рівні для більш ефективного визначення, проектування, створення, тестування та розгортання складних розподілених систем та застарілих технологій інтеграції. Вони потребують підтримки для зберігання, пошуку та опису програмних компонентів та визначень спільного інтерфейсу. Складні технології користувацького інтерфейсу, специфічні для домену, повинні бути спроектовані, впроваджені та перевірені належним чином. Кінцевим користувачам можуть навіть знадобитися досить складні засоби для розширення та вдосконалення своїх додатків після реалізації процесів тестування та налагодження. Керування великими артефактами розробки системного програмного забезпечення має вирішальне значення: командам розробників потрібна підтримка версій, управління конфігураціями, документування та повторне використання величезної кількості артефактів зі складними взаємозалежностями [8].

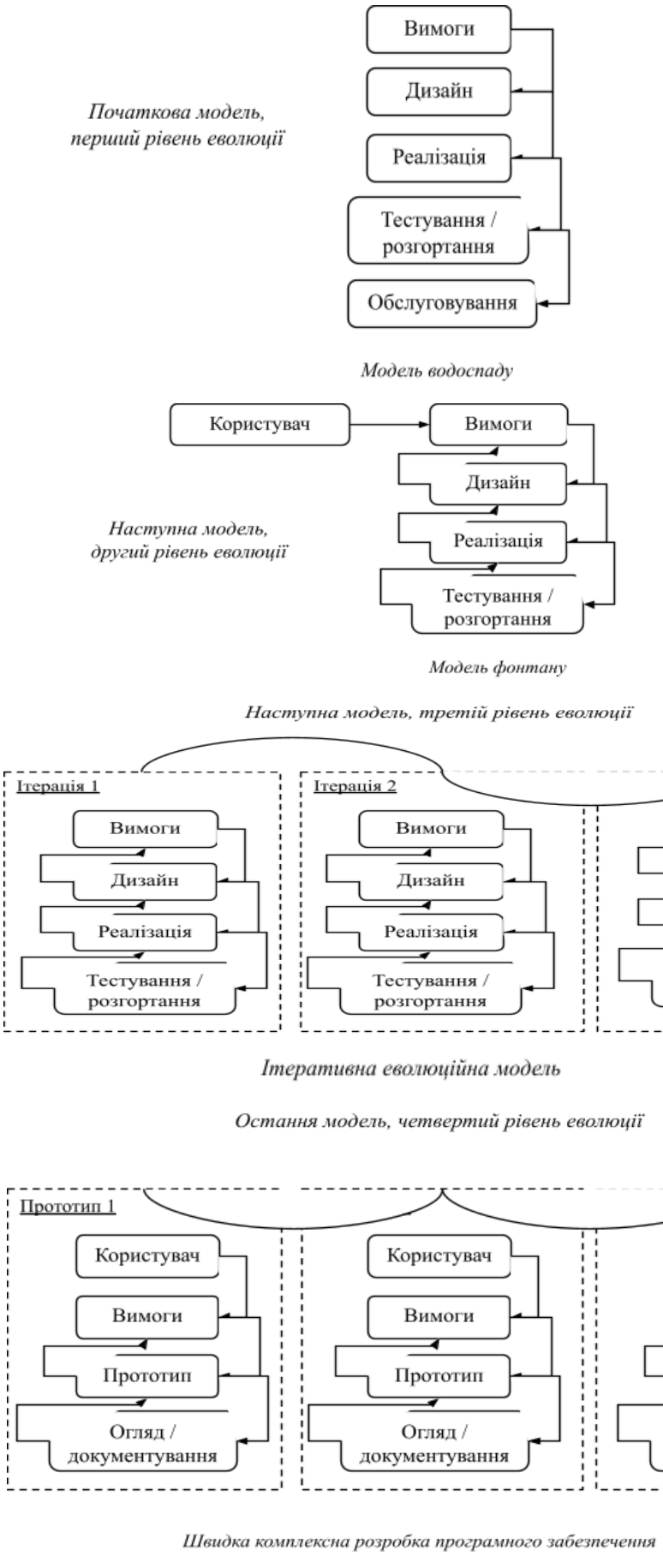

Рис. 1. Еволюція методів та методологій застосовуваних для розробки та підтримки процесів життєвого циклу програмного забезпечення

На сьогодні, практично для кожної фази життєвого циклу розробки програмного забезпечення були розроблені інструменти програмної інженерії. Для багатьох етапів існує велика кількість інструментів, які виконують ті ж або подібні функції. Деякі інструменти надають засоби, які охоплюють багато різних етапів, інші зосереджені на певному виді завдання, технології, мові чи проблемі розробки програмного забезпечення. Програмні засоби характеризуються великою різноманітністю описових термінів. На рисунку 2 наведено загальновизначені етапи життєвого циклу програмного забезпечення та пов'язані з ними категорії інструментів, які зазвичай використовуються на цих етапах розробки. Деякі види програмних засобів мають досить обмежений обсяг, наприклад інструменти тестування, засоби програмування та засоби розробки інтерфейсу користувача. Деякі види інструментів підтримують багато етапів розробки, такі як інструменти RAD, генератори додатків, інструменти CASE та 4GL. Деякі інструменти призначені для використання протягом усього життєвого циклу програмного забезпечення. До них відносяться інструменти управління процесами/проектами, середовища, орієнтовані на процеси та інструменти спільної роботи. Однак вони зазвичай стосуються обмеженої діяльності на кожній фазі і використовуються разом із більш спеціалізованими інструментами. Інструменти підтримки процесів, такі як технологічно-орієнтоване на програмне середовище програмне забезпечення та системи управління робочими процесами, використовуються під час розробки програмного забезпечення для підтримки моделювання, впровадження та вдосконалення програмних процесів [10]. Ці інструменти дозволяють розробникам характеризувати процеси, які вони використовують, «запускати» ці процеси та координувати розробку програмного забезпечення, використовуючи ці процеси. Середовища, орієнтовані на процеси, зазвичай зосереджені на розробці програмного забезпечення, і часто відбувається тісна інтеграція та координація інструментів. Системи управління робочими процесами, як правило, є більш універсальними технологічними засобами, які використовуються для координації широкого спектру робочих процесів, при цьому програмне забезпечення є лише одним із прикладів.

Інструменти архітектури та проектування програмного забезпечення використовуються для моделювання, аналізу та вдосконалення рішень щодо впровадження. Більшість інструментів і методів програмної інженерії для проектування програмного забезпечення надають такі засоби, які зазвичай інтегровані з функціями на рівні специфікації. В останні роки, внаслідок зростання складності більшості системних архітектур, було розроблено різноманітні інструменти специфікації та аналізу архітектури програмного забезпечення. Багато інструментів проектування підтримують функції генерації коду, які використовуються для створення шаблону коду реалізації. Багато з них також надають засоби зворотної інженерії для отримання інформації на рівні проекту з кодових баз. Існує також багато спеціалізованих інструментів зворотного проектування. Деякі працюють над вихідним кодом та відновлюють схеми проектування, інші – над двійковим кодом, відновлюючи вихідний код.

Конструктори додатків зазвичай забезпечують поєднання засобів проектування та впровадження, зазвичай тісно об'єднаних в одному інструменті. Їх часто називають інструментами 4GL (мови четвертого покоління) або генераторами додатків. Багато з цих інструментів орієнтовані на засоби проектування, тісно узгоджені з конкретними мовами впровадження, технологіями та архітектурою.

Інструменти та середовища програмування зосереджені на засобах на рівні реалізації, які зазвичай обмежуються певною мовою та іноді включають дизайнери інтерфейсу користувача, генератори документації, браузери коду та символьні налагоджувачі. Хоча велика кількість асемблерів, компіляторів, редакторів програм та налагоджувачів були розроблені та розгорнуті окремо, протягом багатьох років спостерігається тенденція до включення цих засобів до інтегрованих середовищ розвитку. Існує безліч прикладів, які забезпечують тісно інтегрований генератор програм, редактор, компілятор, засоби налагодження та засоби розгортання. Зараз багато з них включають засоби контролю версій та засоби спільної роботи.

Візуальні мовні системи пропонують альтернативний підхід до програмування та візуалізації програми, наголошуючи на використанні графічних подань програм. Ідея полягає у використанні дво- та тривимірних уявлень, щоб допомогти розробникам краще зрозуміти та побудувати програми. Зазвичай такі системи включають щільно інтегровані візуальні редактори та налагоджувачі, а деякі включають функції моделювання систем на рівні дизайну.

Налагодження, тестування, моніторинг та інші інструменти оцінки використовуються для оцінки якості програмного забезпечення або для допомоги розробникам у виявленні та виправленні помилок у програмному забезпеченні.

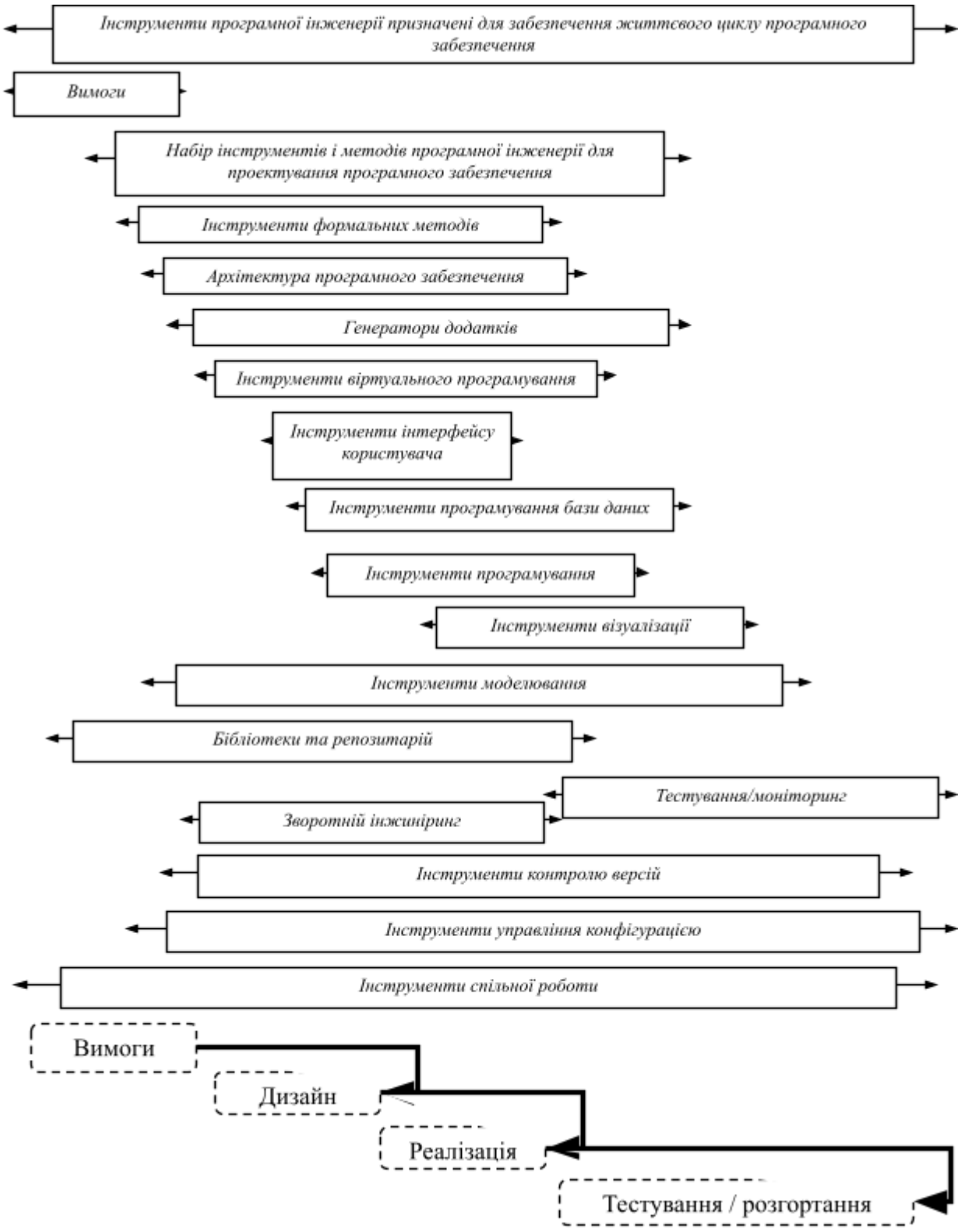

Рис. 2. Інструменти програмної інженерії призначені для забезпечення процесів життєвого циклу програмного забезпечення

Інструменти налагодження зазвичай орієнтовані на низькорівневий аналіз вихідного коду, зазвичай забезпечуючи покроковий, контрольний пункт, моніторинг змінних та просту підтримку візуалізації структури даних. Інструменти анімації алгоритмів та програмної візуалізації забезпечують доступ до програмного алгоритму та структур даних на більш високому рівні за допомогою (як правило) візуальних уявлень про структуру коду, графіків виклику, стилізованих уявлень алгоритмів тощо.

Інструменти контролю версій та конфігурації надають підтримку окремим розробникам або командам розробників для управління своїми артефактами програмного забезпечення у міру їх розвитку. Інструменти версій допомагають керувати створенням, організацією та використанням кількох версій програмних артефактів (традиційно вихідний код, але також можуть включати складені одиниці, проекти та специфікації та документацію). Інструменти документації пропонують підтримку для створення документації користувача за допомогою специфікацій системи та коду. Зазвичай інструменти CASE забезпечують підтримку системної документації (у вигляді аналізу та специфікацій проектування), і хоча деякі пропонують обмежену підтримку документації користувача, спеціалізовані інструменти зазвичай надають більш цільові функції.

Інструменти спільної роботи надають різні функції, що дозволяють групі розробників працювати разом як розподілена команда (з точки зору часу та/або простору). Багато програмних засобів поставляються з (зазвичай обмеженою) підтримкою спільної роботи, але було розроблено багато спеціалізованих інструментів для полегшення роботи в групах. Багато з них, розгорнуті на програмних проектах, не є спеціалізованими для роботи з програмним забезпеченням. Слід зазначити, що деякі описи програмного забезпечення охоплюють багато з перерахованих вище категорій. Наприклад, термін «CASE» зазвичай застосовується до аналізу, проектування та інженерних інструментів, але іноді використовується для позначення всіх інтегрованих програмних засобів, розгорнутих у проекті. Середовища програмної інженерії, інтегровані середовища розробки та інтегровані середовища підтримки проектів – це терміни, приблизно взаємозамінні, які стосуються інструменту або тісно інтегрованого набору інструментів, який підтримує широкий спектр завдань інженерної роботи з програмним забезпеченням. Інтегровані середовища програмування та конструктори додатків зазвичай відносяться до тісно інтегрованих наборів інструментів, які орієнтовані на впровадження, з обмеженими можливостями проектування, документації та тестування. Однак усі ці позначки, що надаються інструментам, не завжди однаково застосовуються продавцями та дослідниками інструментів.

Висновки. У роботі розкрито інструменти програмної інженерії призначені для забезпечення процесів життєвого циклу програмного забезпечення. Оскільки побудова програмних засобів сама по собі є дуже складним завданням інженерії програмного забезпечення, було розроблено багато систем, які спеціально підтримують інженерію програмних засобів. Ці програми зазвичай називають інструментами мета-CASE, конструкторами інструментів або генераторами інструментів. Ці мета інструменти, як правило, підтримують визначення структур даних інструментів та поглядів на структури даних інструментів, надають методи введення/виведення специфікацій та генерують частину або весь код для реалізації зазначеного інструменту.

Перспективи подальших досліджень ґрунтуються на розробці перспективної стратегії інтеграції інструментів програмної інженерії у єдиний комплекс призначений для забезпечення процесів життєвого циклу програмного забезпечення.

Список бібліографічного опису.

1. Козак Е. Программная инженерия (software engineering) – основные тренды развития в условиях глобальной цифровизации // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и Технические Науки. – 2021. – №06. – С. 109-112 DOI 10.37882/2223-2966.2021.06.23
2. Величко, О. М. Особливості процесів життєвого циклу програмного забезпечення для засобів вимірювальної техніки / О. М. Величко, О. В. Грабовський, Т. Б. Гордієнко. Збірник наукових праць Одеської державної академії технічного регулювання та якості [Текст] / [редкол. : Коломієць Л. В. (голова) та ін.]. – Одеса : ОДАТРЯ, 2012 – Вип. 2 (17) 2020. – С .37-45
3. Моренцов Є. І. Система моделей предметної області "Метатехнологія програмування" / Є. І. Моренцов // Проблеми програмування. – 2018. – № 2-3. – С. 236-244. – Режим доступу: http://nbuv.gov.ua/UJRN/Progr_2018_2-3_29.
4. Говорущенко Т.О. Аналіз процесу вибору технології проектування, методології та середовища розроблення програмного забезпечення / Т.О.Говорущенко, Р.А.Малярчук // Вісник Хмельницького національного університету. Технічні науки. – 2014. – №6. – С.186-196.
5. Авраменко А.С., Авраменко В.С., Косенюк Г.В. Тестування програмного забезпечення. Навчальний посібник. – Черкаси: ЧНУ імені Богдана Хмельницького, 2017. – 284 с.
6. Марковець О. В. Формування якісної технічної документації до програмного забезпечення / О. В. Марковець, А. І. Синько // Вісник Вінницького політехнічного інституту. – 2021. – № 2. – С. 98-106. – Режим доступу: http://nbuv.gov.ua/UJRN/vvpi_2021_2_15.

References.

1. Shah, Unnati & Jinwala, Devesh & Patel, Sankita. (2016). An Excursion to Software Development Life Cycle Models: An Old to Ever-growing Models. ACM Transactions on Software Engineering and Methodology. 41. 1-6. 10.1145/2853073.2853080.
2. Kobyliński A., "ISO/IEC 9126 – Analiza modelu jakości produktyw programowych. Prace Naukowe," Akademia Ekonomiczna w Katowicach, Tom: Systemy wspomagania organizacji SWO'2003, 2003, p. 459-468.
3. Basri, Sufyan & Kama, Nazri & Sarkan, Haslina & Ismail, S.A. & Haneem, Faizura. (2016). An Algorithmic-Based Change Effort Estimation Model for Software Development. 177-184. 10.1109/APSEC.2016.034.
4. Bansiya J., and Davis C., "Hierarchical Model for Object-Oriented Quality Assessment," IEEE Transactions on Software Engineering, vol. 28, issue 1, pp. 4-17, 2002.
5. Maznan, Roslinda & Wan Kadir, Wan Mohd Nasir & Kadir, Wan. (2021). A comparative evaluation of the three prominent approaches in adaptable software architecture.
6. Synko A., and Peleshchshyn A., "Software development documentation – documentation types and standards," Scientific Journal of TNTU (Tern.), vol 98, no. 2, pp. 120-128, 2020.
7. Basri, Sufyan & Kama, Nazri & Haneem, Faizura & Ismail, S.A.. (2016). Predicting effort for requirement changes during software development. 380-387. 10.1145/3011077.3011096.
8. Burnstein I. Practical Software Testing. A process-oriented approach. Springer-Verlag, New York, 2003, – 732 p.