УДК 514.182.7

Олександр Мацулевич, кандидат технічних наук, доцент,
доцент кафедри інженерної механіки та комп'ютерного проектування,
Олександр Вершков, кандидат технічних наук, доцент,
доцент кафедри інженерної механіки та комп'ютерного проектування,
Олена Дереза, кандидат технічних наук, доцент, в.о. завідувача
кафедри інженерної механіки та комп'ютерного проектування,
Андрій Чаплінський, старший викладач
кафедри інженерної механіки та комп'ютерного проектування,
Ілля Тетервак, асистент кафедри інженерної механіки
та комп'ютерного проектування,
Таврійський державний агротехнологічний університет
імені Дмитра Моторного, м. Запоріжжя, Україна

ЗАСТОСУВАННЯ CASE-ТЕХНОЛОГІЙ ПРИ КОМП'ЮТЕРНОМУ ПРОЄКТУВАННІ ТЕХНІЧНИХ ОБ'ЄКТІВ І СИСТЕМ

Анотація. В роботі розглянуті питання забезпечення якості проєктів, які пов'язані з процесами верифікації, перевірки та тестування програмного забезпечення. Верифікація – це визначення того, наскільки поточний стан розробки відповідає вимогам певного етапу. Перевірка дає змогу оцінити відповідність результатів розробки початковим вимогам. Частково вона перетинається з тестуванням, яке спрямоване на виявлення відмінностей між фактичними та очікуваними результатами, а також на оцінювання відповідності характеристик програмного забезпечення встановленим вимогам. У процесі реалізації проєкту важливу роль відіграють питання ідентифікації, опису та контролю конфігурації окремих компонентів і всієї системи загалом. Управління конфігурацією є одним із допоміжних процесів, який підтримує основні процеси життєвого циклу програмного забезпечення, насамперед процеси розробки та супроводу ПЗ.

Ключові слова: інформаційна система (ІС), процес верифікації, тестування програмного забезпечення, неоднорідне середовище, апаратна платформа.

Abstract. This paper examines issues related to project quality assurance, specifically those involving software verification, validation, and testing. Verification involves determining the extent to which the current state of development meets the requirements of a specific phase. Validation allows for the assessment of whether the development results conform to the initial requirements. It partially overlaps with testing, which is aimed at identifying differences between actual and expected results, as well as evaluating the compliance of software characteristics with established requirements. During project implementation, issues related to the identification, description, and

control of the configuration of individual components and the entire system as a whole play an important role. Configuration management is one of the supporting processes that underpins the core processes of the software life cycle, primarily software development and maintenance.

Keywords: information system (IS), verification process, software testing, heterogeneous environment, hardware platform.

Сучасний розвиток інформаційних технологій спричиняє постійне ускладнення інформаційних систем (ІС), які створюються в різних сферах економіки [1]. Великі сучасні проєкти ІС зазвичай мають низку характерних рис:

- складність опису систем через значну кількість функцій, процесів, елементів даних та складних взаємозв’язків між ними, що потребує детального аналізу і моделювання;
- наявність великої кількості взаємопов’язаних компонентів і підсистем, кожна з яких виконує власні завдання та має окремі цілі функціонування [4]. Це можуть бути як програми для обробки транзакцій і виконання регламентованих операцій, так і аналітичні системи підтримки прийняття рішень, що працюють із великими обсягами даних;
- відсутність готових аналогів, через що використання стандартних проєктних рішень і типових прикладних систем є обмеженим [5];
- потреба в інтеграції вже існуючих програм із новими програмними розробками [1];
- функціонування систем у неоднорідному середовищі з використанням різних апаратних платформ;
- роз’єднаність і різний рівень підготовки команд розробників, а також відмінності у використанні інструментальних засобів та методів роботи [3];
- значна тривалість реалізації проєкту, що пояснюється як масштабністю організації-замовника, так і обмеженими можливостями команди розробників та різною готовністю окремих підрозділів до впровадження інформаційної системи.

Для успішного створення проєкту ІС необхідно насамперед правильно й точно описати об’єкт проєктування, а також побудувати повні та узгоджені функціональні й інформаційні моделі системи. Досвід розробки ІС свідчить, що це складний, трудомісткий і тривалий процес, який потребує високого рівня професійної підготовки спеціалістів. Проте раніше проєктування інформаційних систем здебільшого здійснювалося на інтуїтивному рівні з використанням неформалізованих методів, що базувалися на практичному досвіді, мистецтві розробників, експертних оцінках і дорогих експериментальних перевірках роботи системи. Крім цього, у процесі створення та використання ІС вимоги й потреби користувачів могли змінюватися або уточнюватися, що ще більше ускладнювало розробку та супровід систем.

У 70-80-х роках під час створення інформаційних систем широко використовувалася структурна методологія, яка забезпечувала розробників чіткими формалізованими методами опису ІС і технічних рішень [6]. Її основою було використання наочних графічних засобів – схем і діаграм, за допомогою яких описувалися різні моделі системи. Завдяки зрозумілості та чіткості структурного аналізу розробники й майбутні користувачі могли брати участь у створенні системи, обговорювати та узгоджувати основні технічні рішення ще на початкових етапах роботи.

Проте широке застосування цієї методології на практиці траплялося нечасто, оскільки без автоматизованих засобів реалізувати її було дуже складно. У ручному режимі важко створювати й графічно оформлювати точні формальні специфікації системи, перевіряти їх на повноту та узгодженість, а також вносити зміни [6]. Навіть якщо вдавалося підготувати якісну систему проєктної документації, її переробка при суттєвих змінах ставала майже неможливою. Через це ручне проєктування часто призводило до таких проблем:

• неточне або неповне визначення вимог до системи;

• неможливість своєчасно виявляти помилки в проєктних рішеннях;

• низька якість документації, що погіршувала експлуатацію системи;

• тривалий процес тестування та незадовільні результати перевірок.

З іншого боку, розробники ІС історично завжди стояли останніми в ряду тих, хто використовував комп'ютерні технології для підвищення якості, надійності і продуктивності у своїй власній роботі (феномен "чоботаря без чобіт").

Саме ці фактори стали причиною появи спеціального класу програмно-технологічних засобів – CASE-засобів, які реалізують CASE-технологію створення та супроводу інформаційних систем. Термін CASE (Computer Aided Software Engineering) сьогодні використовується у значно ширшому значенні, ніж раніше. Спочатку він стосувався лише автоматизації розробки програмного забезпечення, однак згодом почав охоплювати весь процес створення складних інформаційних систем.

Нині CASE-засобами називають програмні продукти, що підтримують усі етапи створення та супроводу ІС: аналіз і визначення вимог, проєктування програмного забезпечення та баз даних, генерацію програмного коду, тестування, документування, контроль якості, управління конфігураціями й проєктом, а також інші процеси розробки. Разом із системним програмним забезпеченням і технічними засобами CASE-засоби формують повноцінне середовище для створення інформаційних систем.

CASE-технологія являє собою методологію проектування ІС, а також набір інструментальних засобів, що дозволяють в наочній формі представити процес проектування.

Появі CASE-технологій передували численні дослідження у сфері методології програмування. Програмування почало набувати рис системного підходу завдяки розвитку мов високого рівня, методів структурного й модульного програмування, мов проєктування та засобів їх підтримки, а також формальних і неформальних мов опису системних вимог і специфікацій.

Крім того, розвитку CASE-технологій сприяли такі чинники:

• підготовка аналітиків і програмістів, які добре володіли принципами структурного та модульного програмування;

• активне впровадження комп'ютерної техніки та постійне зростання її продуктивності, що дало можливість використовувати ефективні графічні засоби й автоматизувати більшість етапів проєктування [2];

• розвиток мережевих технологій, які дозволили об'єднати роботу різних виконавців у єдиний процес проєктування за допомогою спільної бази даних із необхідною інформацією про проєкт [7].

Сучасні CASE-засоби охоплюють велику область підтримки численних технологій проектування ІС: від простих засобів аналізу і документування до повномасштабних засобів автоматизації, що покривають весь життєвий цикл ПЗ [3].

Найскладнішими та найбільш трудомісткими етапами створення інформаційних систем є аналіз і проєктування [4]. Саме на цих етапах CASE-засоби допомагають забезпечити високу якість технічних рішень і підготовку необхідної проєктної документації. Важливе значення при цьому мають методи візуального подання інформації. Вони передбачають створення структурних та інших видів діаграм у реальному масштабі часу, використання широкої кольорової палітри та автоматичну перевірку правильності побудови моделей. Графічні засоби моделювання предметної області дають змогу розробникам у наочній формі аналізувати вже існуючі інформаційні системи, удосконалювати їх відповідно до поставлених завдань і враховувати наявні обмеження.

До CASE-засобів належать як відносно недорогі програми для персональних комп'ютерів із обмеженим набором функцій, так і потужні дорогі системи, розраховані на роботу в неоднорідних обчислювальних середовищах і на різних апаратних платформах. Сьогодні ринок програмного забезпечення налічує близько 300 різновидів CASE-засобів, а найсучасніші з них активно використовуються провідними світовими компаніями.

Зазвичай CASE-засобами вважають програмні продукти, які автоматизують певні процеси життєвого циклу програмного забезпечення та мають такі основні характеристики: потужні графічні інструменти для опису й документування інформаційних систем, що забезпечують зручну взаємодію з розробником і підтримують його творчі можливості; інтеграцію окремих компонентів CASE-засобів, яка дозволяє ефективно керувати процесом створення інформаційної системи; використання спеціально організованого сховища проєктних даних і метаданих – репозиторію.

Інтегрований CASE-засіб або комплекс засобів, що підтримує повний життєвий цикл програмного забезпечення, зазвичай містить такі компоненти:

• репозиторій – основний елемент CASE-засобу, який забезпечує збереження версій проєкту та його складових, синхронізацію роботи різних розробників під час колективної роботи, а також перевірку даних на повноту й узгодженість;

• графічні засоби аналізу та проєктування, що дозволяють створювати й редагувати взаємопов’язані діаграми та моделі інформаційної системи;

• засоби розробки програмних додатків, зокрема мови четвертого покоління (4GL) і генератори програмного коду;

• засоби конфігураційного управління;

• засоби створення документації;

• засоби тестування програмного забезпечення;

• засоби управління проєктом;

• засоби реінжинірингу для модернізації та вдосконалення існуючих систем.

Усі сучасні CASE-засоби можна класифікувати за різними типами та категоріями. Класифікація за типами відображає функціональне призначення CASE-засобів і їх орієнтацію на певні процеси життєвого циклу системи. Класифікація за категоріями визначає рівень інтеграції виконуваних функцій і охоплює окремі локальні засоби, що вирішують невеликі самостійні завдання, комплекси частково інтегрованих засобів, які підтримують більшість етапів

життєвого циклу ІС, а також повністю інтегровані системи, що забезпечують підтримку всього життєвого циклу ІС і працюють на основі спільного репозиторію.

Крім цього, CASE-засоби можна класифікувати за такими ознаками:

• використовуваними методологіями та моделями систем і баз даних;

• рівнем інтеграції із системами управління базами даних (СУБД);

• доступними апаратними та програмними платформами.

Класифікація за типами переважно відповідає компонентному складу CASE-засобів і включає такі основні категорії:

• засоби аналізу (Upper CASE), які призначені для створення та дослідження моделей предметної області (Design/IDEF (Meta Software), BPwin (Logic Works));

• засоби аналізу та проєктування (Middle CASE), що підтримують найпоширеніші методології проєктування та застосовуються для формування проєктних специфікацій (Vantage Team Builder (Cayenne), Designer/2000 (ORACLE), Silverrun (CSA), PRO-IV (McDonnell Douglas), CASE.Аналітик (МакроПроджект)).

Результатом використання таких засобів є створення специфікацій компонентів та інтерфейсів системи, розробка архітектури, алгоритмів і структур даних.

• засоби проєктування баз даних, які забезпечують моделювання даних і автоматичне створення схем баз даних, зазвичай мовою SQL, для найпоширеніших СУБД. До таких засобів належать ERwin (Logic Works), S-Designor (SDP) і DataBase Designer (ORACLE). Засоби для проєктування баз даних також входять до складу CASE-засобів Vantage Team Builder, Designer/2000, Silverrun і PRO-IV;

• засоби розробки додатків. До цієї категорії належать засоби 4GL, зокрема Uniface (Compuware), JAM (JYACC), PowerBuilder (Sybase), Developer/2000 (ORACLE), New Era (Informix), SQL Windows (Gupta), Delphi (Borland) та інші, а також генератори програмного коду, що є складовою Vantage Team Builder, PRO-IV і частково Silverrun [2];

• засоби реінжинірингу, призначені для аналізу програмного коду та схем баз даних із подальшим створенням на їх основі різних моделей і проєктних специфікацій. Інструменти аналізу схем баз даних і побудови ERD входять до складу Vantage Team Builder, PRO-IV, Silverrun, Designer/2000, ERwin і S-Designor. У сфері аналізу програмного коду найбільшого поширення набули об'єктно-орієнтовані CASE-засоби, які забезпечують реінжиніринг програм, написаних мовою C++ (Rational Rose (Rational Software), Object Team (Cayenne)).

Допоміжні типи включають: засоби планування і управління проектом (SE Companion, Microsoft Project та ін.); кошти конфігураційного управління (PVCS (Intersolv)); кошти тестування (Quality Works (Segue Software)); кошти документування (SoDA (Rational Software)).

На сьогодні ринок програмного забезпечення містить наступні розвинені CASE-засоби: Vantage Team Builder (Westmount I-CASE); Designer / 2000; Silverrun; ERwin + BPwin; S-Designor; CASE.Аналітік.

Крім того, на ринку постійно з'являються як нові для вітчизняних користувачів системи (наприклад, CASE / 4/0, PRO-IV, System Architect, Visible Analyst Workbench, EasyCASE), так і нові версії і модифікації перерахованих систем.

Одним із ключових понять методології проєктування інформаційних систем є життєвий цикл програмного забезпечення (ЖЦ ПЗ). Життєвий цикл ПЗ – це безперервний процес, який починається з моменту прийняття рішення про створення програмного забезпечення та завершується його повним виведенням з експлуатації.

Основним нормативним документом, який регламентує життєвий цикл програмного забезпечення, є міжнародний стандарт ISO/IEC 12207 (ISO – International Organization of Standardization, Міжнародна організація зі стандартизації; IEC – International Electrotechnical Commission, Міжнародна електротехнічна комісія). Цей стандарт визначає структуру життєвого циклу, яка

включає процеси, дії та завдання, що повинні виконуватися під час створення програмного забезпечення.

Структура ЖЦ ПЗ відповідно до стандарту ISO/IEC 12207 базується на трьох основних групах процесів:

• основні процеси життєвого циклу ПЗ (придбання, постачання, розробка, експлуатація та супровід);

• допоміжні процеси, які забезпечують підтримку основних процесів (документування, управління конфігурацією, забезпечення якості, верифікація, атестація, оцінювання, аудит і розв'язання проблем);

• організаційні процеси (управління проєктами, створення інфраструктури проєкту, визначення, оцінка й удосконалення життєвого циклу, а також навчання персоналу).

Розробка охоплює всі роботи, пов'язані зі створенням програмного забезпечення та його компонентів відповідно до встановлених вимог. До цього входять оформлення проєктної й експлуатаційної документації, підготовка матеріалів для перевірки працездатності та якості програмних продуктів, а також матеріалів для навчання персоналу. Зазвичай процес розробки ПЗ включає аналіз, проєктування та програмування.

Експлуатація передбачає впровадження компонентів програмного забезпечення в робоче середовище, налаштування баз даних і робочих місць користувачів, забезпечення необхідною документацією та проведення навчання персоналу [7]. Крім того, вона охоплює безпосереднє використання системи, виявлення та усунення проблем, модифікацію програмного забезпечення відповідно до встановлених правил, а також підготовку пропозицій щодо вдосконалення й модернізації системи.

Управління проєктом пов'язане з плануванням і організацією робіт, формуванням команди розробників та контролем термінів і якості виконання завдань [8]. Технічне й організаційне забезпечення проєкту включає вибір методів

та інструментальних засобів реалізації, визначення способів опису проміжних етапів розробки, створення методів і засобів тестування програмного забезпечення, а також навчання персоналу.

Забезпечення якості проєкту пов’язане з процесами верифікації, перевірки та тестування програмного забезпечення. Верифікація – це визначення того, наскільки поточний стан розробки відповідає вимогам певного етапу. Перевірка дає змогу оцінити відповідність результатів розробки початковим вимогам. Частково вона перетинається з тестуванням, яке спрямоване на виявлення відмінностей між фактичними та очікуваними результатами, а також на оцінювання відповідності характеристик програмного забезпечення встановленим вимогам.

У процесі реалізації проєкту важливу роль відіграють питання ідентифікації, опису та контролю конфігурації окремих компонентів і всієї системи загалом. Управління конфігурацією є одним із допоміжних процесів, який підтримує основні процеси життєвого циклу програмного забезпечення, насамперед процеси розробки та супроводу ПЗ.

При створенні проектів складних ІС, що складаються з багатьох компонентів, кожен із яких може мати різні варіанти або версії, виникає проблема обліку їхніх зв’язків і функцій, формування уніфікованої структури та забезпечення розвитку всієї системи. Управління конфігурацією дозволяє організувати, систематично враховувати й контролювати внесення змін у ПЗ на всіх стадіях ЖЦ. Загальні принципи та рекомендації щодо конфігураційного обліку, планування і управління конфігураціями ПЗ відображені у стандарті ISO 12207-2.

Кожен процес характеризується певними завданнями, методами їх виконання, вхідними даними, отриманими на попередньому етапі, а також результатами роботи. Зокрема, результатами аналізу є функціональні моделі, інформаційні моделі та відповідні діаграми. ЖЦ ПЗ має ітераційний характер,

тобто результати чергового етапу часто спричиняють зміни в проєктних рішеннях, прийнятих на попередніх етапах.

Сутність структурного підходу до розробки ІС полягає в її декомпозиції, тобто поділі на автоматизовані функції: система поділяється на функціональні підсистеми, які далі розбиваються на підфункції, завдання та окремі процедури. Процес декомпозиції триває до рівня конкретних процедур. При цьому автоматизована система зберігає цілісне уявлення, у якому всі компоненти взаємопов'язані. Якщо ж розробка здійснюється «знизу-вгору», від окремих завдань до всієї системи, цілісність втрачається, що створює проблеми під час інтеграції окремих компонентів.

Усі найпоширеніші методології структурного підходу базуються на низці загальних принципів. Основними базовими принципами є:

• принцип «розділяй і володарюй» – спосіб вирішення складних проблем шляхом їх поділу на менші незалежні завдання, які легше зрозуміти та розв'язати;

• принцип ієрархічного впорядкування – організація складових частин проблеми у вигляді ієрархічних деревоподібних структур із поступовим додаванням нових деталей на кожному рівні.

Виділення лише двох базових принципів не означає, що інші є менш важливими, оскільки нехтування будь-яким із них може призвести до серйозних наслідків, включаючи провал усього проекту. До основних принципів також належать:

• принцип абстрагування – виділення суттєвих характеристик системи та відокремлення від несуттєвих деталей;

• принцип формалізації – необхідність застосування чіткого методичного підходу до вирішення поставлених завдань;

• принцип несуперечності – забезпечення обґрунтованості та узгодженості всіх елементів системи;

• принцип структурування даних – дані повинні бути впорядкованими та організованими за ієрархічним принципом.

У структурному аналізі переважно використовуються дві групи засобів, які відображають функції, та взаємозв'язки між даними. Кожній із цих груп відповідають певні типи моделей і діаграм, серед яких найпоширенішими є:

• SADT (Structured Analysis and Design Technique) – моделі та відповідні функціональні діаграми.

• DFD (Data Flow Diagrams) діаграми потоків даних;

• ERD (Entity-Relationship Diagrams) діаграми "сутність-зв'язок".

На стадії проектування ІС моделі розширюються, уточнюються і доповнюються діаграмами, що відбивають структуру програмного забезпечення: архітектуру ПЗ, структурні схеми програм і діаграми екранних форм.

Перераховані моделі в сукупності дають повний опис ІС незалежно від того, чи є вона існуючої чи знову розробляється. Склад діаграм у кожному конкретному випадку залежить від необхідної повноти опису системи.

Список використаних джерел

1. Мацулевич О. Є. Застосування спеціалізованої PLM-системи Technologi CS при розробці автоматизованої системи ведення конструкторсько-технологічних баз даних підприємства сільськогосподарського машинобудування. *Праці Таврійського державного агротехнологічного університету*. 2024. Вип. 24, т. 1 https://doi.org/10.32782/2078-0877-2024-24-1-13
2. Мацулевич О. Є., Гавриленко Є. А., Мірошниченко М. Ю., Гешева Г. В. Набуття навичок комп'ютерної обробки аудіо сигналів з використанням програмного забезпечення Adobe Audition. *Розвиток сучасної науки та освіти :* матеріали IV Міжнародної наук.-практ. Інтернет-конф. (Запоріжжя, 29-31 травня 2023 р.). Запоріжжя : ТДАТУ, 2023. С. 80-86.
3. Вершков О. О., Мацулевич О. Є., Дереза О. О. Загальні налаштування системи MASTERCAM для виконання завдань з розробки управляючих програм токарної обробки валів. *Розвиток сучасної науки та освіти: реалії, проблеми якості, інновації* : матеріали V Міжнародної наук.-практ. інтернет-конф. (м. Запоріжжя, 29-31 травня 2024 р.). Запоріжжя : ТДАТУ, 2024. С. 100-105.
4. Alrefo I. F., Rawashdeh M. O., Matsulevych O., Vershkov O., Halko S., Suprun O. Designing the functional surfaces of camshaft cams of internal combustion engines.

Naukovyi Visnyk Natsionalnoho Hirnychoho Universytetu. 2024. Vol. 3. P. 72–78. https://doi.org/10.33271/nvngu/2024-3/072
5. Alrefo I. F., Matsulevych O., Vershkov O., Halko S., Suprun O., Miroshnyk, O. Designing the working surfaces of rotary planetary mechanisms. *Naukovyi Visnyk Natsionalnoho Hirnychoho Universytetu*. 2023. Vol. 4. P. 82-88. https://doi.org/10.33271/nvngu/2023-4/082. ISSN 2071-2227
6. Паляничка Н. О., Вершков О. О., Антонова Г. В. Аналіз новітніх пристроїв для гомогенізації. *Праці Таврійського державного агротехнологічного університету*. 2017. Вип. 17, т. 3. С. 194-200.
7. Мацулевич О. Є., Гавриленко Є. А., Мірошниченко М. Ю., Гешева Г. В. Набуття навичок комп'ютерної обробки аудіо сигналів з використанням програмного забезпечення Adobe Audition. *Розвиток сучасної науки та освіти* : матеріали IV Міжнародної наук.-практ. Інтернет-конф. (Запоріжжя, 29-31 травня 2023 р.). Запоріжжя : ТДАТУ, 2023. С. 80-86.
8. Вершков О. О., Мацулевич О. Є., Дереза О. О. Загальні налаштування системи MASTERCAM для виконання завдань з розробки управляючих програм токарної обробки валів. *Розвиток сучасної науки та освіти: реалії, проблеми якості, інновації :* матеріали V Міжнародної наук.-практ. інтернет-конф. (м. Запоріжжя, 29-31 травня 2024 р.). Запоріжжя : ТДАТУ, 2024. С. 100-105.