https://doi.org/10.31891/2307-5732-2025-357-48
УДК 004.91

ОРЛОВСЬКИЙ СЕРГІЙ
Київський національний університет технологій та дизайну
https://orcid.org/0009-0008-3037-7999
e-mail: 0734666127o@gmail.com
СТАЦЕНКО ДМИТРО
Київський національний університет технологій та дизайну
https://orcid.org/0000-0002-3064-3109
e-mail: statsenko.dv@knutd.edu.ua

ЖИТТЕВИЙ ЦИКЛ РОЗРОБКИ СИСТЕМИ ВЕДЕННЯ ОБЛІКУ В ЗБРОЙНИХ СИЛАХ УКРАЇНИ

В досліджені детально розглянуті існуючі моделі життєвого циклу розробки програмного забезпечення. Виявлено переваги і недоліки існуючих рішень для процесу розробки. Також детально описано всі процеси які задіяні в розробці.

Запропоновано комбінований підхід для реалізації процесу розробки системи обліку в Збройних силах України. схематично показано всі етапи створення програмного забезпечення і їх взаємодія між собою. Також визначені основні проблеми що виникають на етапах розробки, та запропоновано шляхи їх вирішення.

Дане дослідження допомагає обрати сучасні підходи для створення і розробки додатків. Також представлено створення робочого прототипу, дослідної експлуатації і процес інтеграції. Всі 3 модуля виділені окремо для більш детального огляду всіх етапів створення продукту.

***Ключові слова**: життєвий цикл, програмне забезпечення, модуль*

ORLOVSKYI SERHII, STATSENKO DMYTRO
Kyiv National University of Technologies and Design

LIFE CYCLE OF DEVELOPMENT OF A RECORD-KEEPING SYSTEM IN ARMED FORSES OF UKRAINE

The article addresses the problem of selecting and adapting appropriate software development life cycle (SDLC) models for creating a digital record-keeping system in the Armed Forces of Ukraine. The research provides a comprehensive analysis of classical and modern SDLC models, including Waterfall, V-model, Incremental, Iterative, Spiral, Agile, and Scrum, with a focus on their advantages, limitations, and applicability to defense-related information systems. It highlights that an inappropriate methodological choice can negatively affect cost, flexibility, and functionality, emphasizing the importance of combining different approaches depending on the project phase and specific requirements. The authors propose a hybrid model that integrates elements of Waterfall and Iterative methodologies, enabling the development of a functional minimum viable product (MVP) with clear requirements and gradual expansion of features through iterative testing and refinement. Special attention is given to the challenges of transparency, traceability, monitoring, and uncontrolled changes during the development process, as well as to the complexity of error detection and debugging. The study also considers the integration of the accounting system with existing platforms such as ARMY+, outlining the need for a dedicated development team, well-structured documentation, and continuous testing at every stage. The proposed approach ensures flexibility in meeting evolving user needs while maintaining strict control over project planning and execution. A prototype, experimental deployment, and integration processes are described as distinct modules to provide a detailed view of the product's life cycle. The findings of the research underline that hybrid methodologies are particularly suitable for military applications, where stability, adaptability, and reliability must be balanced. Ultimately, the study contributes to the methodological foundation for designing record-keeping systems in the defense sector and offers practical recommendations for their implementation in the Ukrainian Armed Forces.

***Keywords**: life cycle, software, module*

Стаття надійшла до редакції / Received 21.07.2025
Прийнята до друку / Accepted 15.08.2025

Постановка проблеми

Сьогодні, у світі технологій, програмне забезпечення є невідємною частиною усіх сфер діяльності, від медицини до виробництва. Без інформаційних технологій не може існувати жодна установа чи то комерційна чи державна. Так само це стосується і збройних сил України. Виходячи з висновків попереднього дослідження можливостей автоматизації ведення обліку в збройних силах України [1] доцільною є розробка вітчизняної систему електронного обліку. Вирішення цієї задачі зумовлює появу багатьох технічних питань, зокрема: вибір бази даних, мови програмування, додаткового програмного забезпечення, організації процесу розробки.

Оскільки вибір життєвого циклу розробки є першим етапом створення системи, розгляне відомі сьогодні моделі його організації::

- каскадна або водоспадна модель (Waterfall model) […] – фази проекту починаються одна за одною;

- V-подібна модель (V-model) […] – відрізняється від каскадної повним контролем і на стадії написання вимог вже починається тестування;

- Інкрементна модель (Incremental model) […] –програмне забезпечення розроблюється з лінійною послідовністю стадій, але в декілька версій;

- Спіральна модель (Spiral model) […] – життєвий шлях продукту можна представити у вигляді спіралі, що починається на етапі планування і розкручується з кожною новою стадією;

- Гнучка модель (Agile model) […] – являє собою сукупність різних підходів до розробки програмного забезпечення, що забезпечує можливість зміни процесу розробки в ході реалізації проекту, адаптацію до можливої зміни вимог;

- Методологія Скрам (Scrum) […] – робить акцент на якісному контролі процесу розробки.

Невдалий вибір методології може негативно вплинути на всі показники програмного забезпечення, а саме гнучкість, вартість і функціональність. Гнучкі методології розробки допомагають вирішити основні проблеми. Дуже часто виникають випадки коли найефективнішим буде застосування гібридних методології у зв'язці, наприклад, Agile та каскадна модель.

Також треба звернути увагу на інші проблеми, що виникатимуть в процесі розробки програмного забезпечення, а саме: прозорість, коли в будь-який момент часу складно сказати на якій стадії перебуває проєкт, що є наслідком недостатнього планування структури чи архітектури майбутнього продукту; контроль, коли без чіткої оцінки стадій розробки зриваються плани та графіки виконання робіт і перевищуються встановлені бюджети; проблема трасування, коли інформація про стан та хід виконання записується в логах та використовується для діагностики типових проблем, що виникають під час розробки системи; недоліки моніторингу, що дозволяє контролювати хід розробки в реальному часі; неконтрольовані зміни, що виникають коли у споживачів з’являються нові ідеї щодо роботи того чи іншого компоненту.

Пошук та виправлення помилок в розробці це дуже складний процес. Оскільки число помилок у програмах завжди невідомо з самого початку, то відповідно невідома і тривалість налагодження програм і вкрай складно гарантувати відсутність помилок в програмах.

Аналіз попередніх досліджень

У роботі […]автори визначають основні проблеми, що впливають на розробку програмного забезпечення (ПЗ) :

- апаратна складність випереджає наше вміння створювати ПЗ, що використовує потенційні можливості апаратури;

- наше вміння створювати нові програми відстає від вимог до нових програм;

- нашим можливостям експлуатувати існуючі програми загрожує низька якість їх розробки. Ключем до вирішення цих проблем є грамотна організація процесу створення ПО, реалізація технологічних принципів промислового конструювання програмних систем (ПС)[5].

Також варто звернути увагу на існуюче програмне забезпечення. Для прикладу SAP SE написана мовою ABAP/4 ((англ. Advanced Business Application Programming - пропрієтарна внутрішня мова програмування високого рівня з коболоподібним синтаксисом, що використовується в додатках корпорації SAP). Реалізує роботу з внутрішніми структурами даних, інтерфейсом користувача SAP R/3, транзакціями, звітами, інтерфейсами завантаження та вивантаження даних. Використовується виключно для бізнес-додатків та проміжного програмного забезпечення компанії SAP. Підтримує об'єктно-орієнтоване програмування. Має збирач сміття. Вихідний текст ABAP компілюється («генерується») в ABAP-байт-код («report load»), що запускається у спеціалізованому середовищі виконання [6].

Але як зазначалося в досліджені можливостей автоматизації ведення обліку в Збройних силах України [1] SAP SE має певні недоліки серед яких є ціна. Можливою альтернативою є програмний продукт BAS від польської компанії «NetHelp», який за своєю суттю є наближеним до 1С (заборонена для використання) для українського ринку. Цей продукт має вбудовану мову програмування BAS. Середовище виконання мови - це платформа. Візуальне середовище розробки - це конфігуратор. Версії мови зворотно-сумісні одна з одною [7].

Платформою надається фіксований набір базових класів, орієнтованих на рішення типових задач прикладної області:

- Константа.
- Довідник.
- Документ.
- Журнал документів.
- Перечислення.
- Звіт.
- Обробка.
- План рахунків.
- Регістри відомостей.
- Регістри накопичень та ін.

На підставі цих класів можна створити будь-яку кількість дочірніх класів одного класу, тобто створити скільки завгодно довідників

Виклад матеріалу дослідження

Розробка програмного забезпечення є багатогранним складним процесом створення додатків та програм. Вона охоплює всі етапи створення програми: планування, проектування, тестування та підтримку. цей процес може включати як розробку невеликих додатків так і великих систем для бізнесу та установ.

Програмне забезпечення можна поділити на три основні категорії:

- системне, тут будуть операційні системи, драйвери, що керують апаратною частиною ЕОМ.
- прикладне, що виконує конкретні задачі типу SAP, Word.
- вбудоване, для автомобілів, техніки.

Розглянемо водоспадну модель створення програмного забезпечення (рис. 2), що виникла та була формально описана однією з найперших. Її основними етапами є:

1. Аналіз вимог. На цій стадії треба зібрати всі завдання які має виконувати програма та проаналізувати вимоги яким вона має відповідати.
2. Проектування архітектури програми, де ми визначаємо основні структури і компоненти, що вирішують поставлені задачі.
3. Стадія програмування. Це етап на якому пишеться код на обраній мові програмування, інтегруються різні компоненти та бібліотеки.
4. Тестування. Цей процес гарантує, що програма працює належним чином вирішує поставлені задачі, відповідає вимогам, які були поставлені на етапі аналізу вимог, не містить помилок.
5. Впровадження та розгортання – передача програми користувачам.
6. Підтримка, в процесі якої здійснюється оновлення програми додавання нових функцій, зміна застарілих рішень.

Рис.1. Каскадна модель життєвого циклу програмного забезпечення

Серед плюсів каскадного підходу потрібно виділити повне документування всіх процесів, чітке планування термінів, та прозорість всіх процесів. А до мінусів можна віднести необхідність затвердження повного списку вимог вже на першому етапі, та збільшення часу розробки в разі необхідності внесення змін.

Розглянемо V-подібну модель, що є покращеною версією каскадної моделі. Тут ми на кожному етапі контролюємо процес щоб в впевнитися, що ми готові перейти до наступного етапу. Специфічним є те що тестування починається ще на рівні написання вимог, і з переходом на нові рівні кількість тестування збільшується.

Дану модель зазвичай використовують коли проєкт має суттєві обмеження у часі та вимагає постійного тестування. Звідси випливають переваги та недоліки використання такого підходу. Серед переваг ми визначимо строгу етапність, чіткий тайм-менеджмент та тестування на всіх рівнях. До недоліків відносяться недостатня гнучкість, недостатній аналіз ризиків, відсутність паралельної роботи і власне створення програми на етапі написання коду.

Рис 2. V-подібна модель життєвого циклу розробки ПЗ

Наступною розглянемо ітеративну модель. У спрощеному вигляді, ітеративна модель складається з чотирьох основних стадій, які повторюються у кожній з ітерацій (plan-do-check-act).

Рис. 3. Ітеративна модель розробки ПЗ

Як і попередні моделі ітеративна також має свої переваги та недоліки. До переваг відноситься гнучкість моделі, працююча програма на початковій стадії, аналіз ризиків для кожної ітерації окремо. Недоліки –це проблеми з загальною архітектурою, що виникають через те, що не всі вимоги відомі на початковому етапі.

Наступною моделлю процесу розробки ПЗ є спіральна модель. Особливість її представлення полягає у створені площини розбитої на 4 секції кожна з яких представляє окремий етап розробки ПЗ.

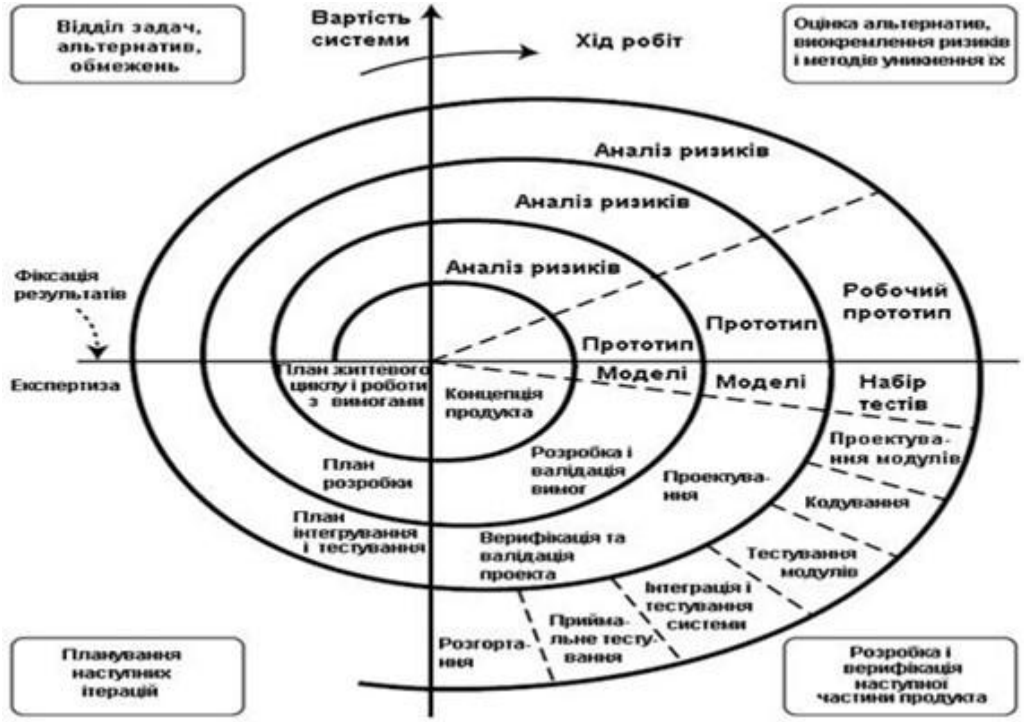

Рис. 4. Спіральна модель розробки ПЗ

Особливостями даної моделі є гнучкість, покращений аналіз ризиків, якісна документація та раннє створення робочого прототипу, але й недоліком є висока вартість використання , що не підійде для невеликого проекту.

Скрам (Scrum) — це не лінійний підхід до гнучкої розробки ПЗ. За Скрамом, продукт розробляють не відразу весь, а невеликими, готовими до релізу частинами, кожну з яких завершують протягом короткої ітерації або спринту. Перевагами Скраму є швидка розробка, швидкість відгуків на потреби клієнта, зменшення вартості розробки. Але ці переваги зумовлюють появу недоліків, зокрема, відсутність чіткого дедлайну проєкту, складність застосування в невеликих командах, великі шанси провалити проект якщо в команді немає злагодження.

Аналіз розглянутих методів розробки представлено в табл. 1. В залежності від завдань в проекті та технічних можливостей доцільно обирати ту чи іншу модель розробки а іноді та комбінувати їх між собою.

Таблиця 1

Моделі розробки ПЗ

Модель / Функціонал	Каскадна модель	V-подібна модель	Ітеративна модель	Спіральна модель
Аналіз вимог	З самого початку	З самого початку	Кожен етап	З самого початку
Проектування	З самого початку	З самого початку	Кожен етап	Прототип на початку
Програмування	Мала команда	Мала команда	Мала команда	Велика команда
Тестування	Перед впровадженням	Кожен етап	Кожен етап	Кожен етап
Впровадження	Складність змін	Складність змін	Легкі зміни	Легкі зміни

Для розробки системи обліку варто обрати комбінацію каскадної та ітеративної моделей. Їх можливо комбінувати таким чином, щоб на початковому етапі визначити всі вимоги для проекту, архітектуру проектувати для кожного етапу окремо, залучити невелику кількість розробників, і тестувати на кожному етапі що забезпечить легкість підтримки в майбутньому. Схематично вона може виглядати наступним чином:

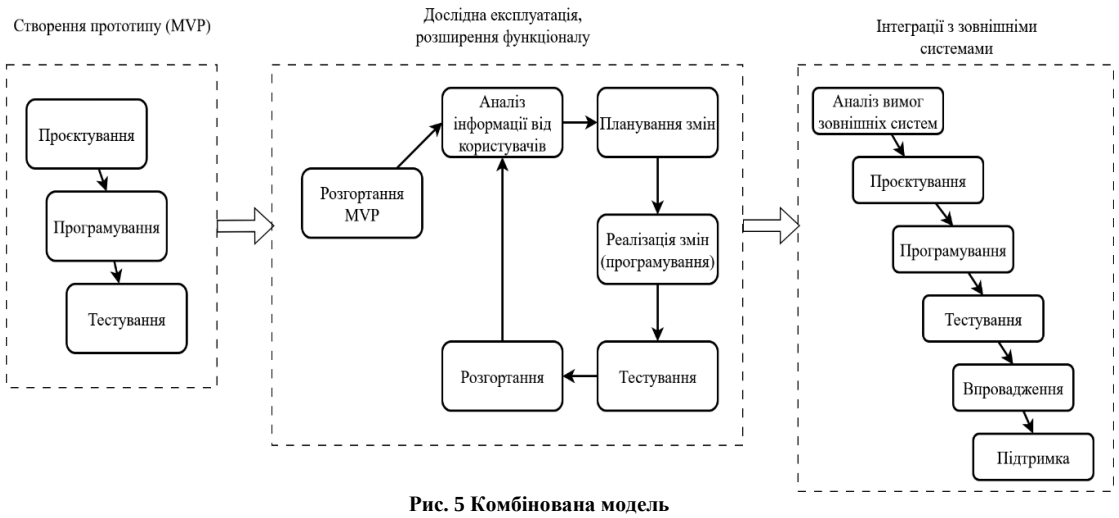

Рис. 5 Комбінована модель

На даній схемі відображено етапи розробки програмного забезпечення при комбінованому підході. Така схема має певні особливості так як тут є можливість контролю кожного етапу розробки, також при необхідності можливе повернення до попереднього етапу. Тут нема чітко визначеного порядку етапів розробки, що дає можливість більш гнучко реагувати на потреби замовника. Для прикладу кожна з запропонованих моделей починається з планування, після цього етапу ми вільні як почати тестування того що заплановано так і зробити аналіз вимог, ми також можемо зразу проектувати наш додаток після планування. Але важливо зазначити що при комбінованій моделі розробки програмного забезпечення мають бути пройдені всі основні етапи. Тобто ми не можемо розробити план, спроектувати наш додаток і одразу його впровадити. Нам потрібно обов'язково визначити вимоги і провести тестування. Що також є особливим це те що впровадження може відбуватися паралельно з тестуванням, це збереже час і ресурси для фінальної версії продукту. На етапі створення прототипу використовуємо скорочену водоспадну модель, оскільки мінімальні вимоги відомі (це мінімальний перелік функцій, які повинні бути реалізовані в обліковій системі). Етапи впровадження та підтримки тут не використовуються оскільки далі ми переходимо до етапу дослідної експлуатації та розширення функціоналу, який організований за іншою моделлю. На етапі дослідної експлуатації та розширення функціоналу використовуємо ітераційну модель. На етапі інтеграції з зовнішніми системами використовуємо стандартну водоспадну модель, оскільки вважаємо, що вимоги зовнішньої системи відомі на початку розробки. Якщо вимоги змінюються, наприклад, через розширення функціоналу зовнішньої системи, запускаємо окремий етап інтеграції (також з водоспадною моделлю).

При виборі моделі розробки ми маємо звертати увагу на зовнішні умови. Якщо ми визначаємо технічні умови перед початком розробки – обираємо каскадну модель, якщо ж нам потрібно постійно перевіряти хід робіт – бажано використовувати модель, що дасть змогу перевіряти кожен етап розробки. Якщо ми обмежені в кількості розробників в команді, або команда недостатньо злагоджена, то модель Скрам буде дуже ризиковано використовувати.

Для системи обліку майна в Збройних силах України також буде специфічним її інтеграція в АРМІЮ +. Тут потрібна окрема команда розробників, чітка постановка задачі, продумана архітектура та документація на кожному етапі. Також особливим при такій умові буде тестування на кожному кроці інтеграції.

Комбінований підхід при розробці додатку для обліку в ЗСУ умовно буде виглядати так:

- Початковий етап: створення MVP (minimal value product), реальних користувачів ще немає, але є чітка вимоги, так як правила ведення обліку для всіх систем однакові. Тут нам підійде каскадна модель;
- Наступним кроком, після запуску MVP переходимо до етапу дослідної експлуатації добавляємо користувачів, які надсилають зауваження/пропозиції. Тут недоліки каскадної моделі стають вагомими тому краще використати ітеративну;
- На етапі інтеграції з АРМІЯ + ми будемо реалізовувати вимоги по каскадній моделі вийде каскадна всередині ітеративної використовувати ітеративну модель так як потрібна додаткова команда яка буде за це відповідати, та спіральна модель забезпечить легкі зміни.

Почнімо ж з першого етапу – аналізу вимог для системи ведення обліку в збройних силах України. Ми будемо мати основні класи вимог: бізнес вимоги, вимоги користувачів, функціональні та нефункціональні вимоги.

- бізнес вимоги – високорівневі цілі замовника, часто можуть бути окремим документом, що містить у собі роз'яснення для чого недохідна ця система, визначає межі проекту це буде так званий статут проєкту. В даному випадку вирішуються задачі відходу від бюрократії, автоматизації системи обліку в збройних силах України, повного перехід від паперового обліку до комп'ютеризації;

- вимоги користувачів – перелік завдань які має вирішувати система. Також у вимогах має бути вказано, що користувач зможе робити з системою. Для прикладу розглянемо облік майна: користувач має добавити номенклатуру з усіма характеристиками (шт., ціна, об'єм, термін придатності та ін.), система має записати дану інформацію в базу даних за заданими параметрами, за необхідності також має повернути інформацію користувачу при опрацюванні відповідних запитів.
- Функціональні вимоги або вимоги поведінки, які визначають функціональність системи в межах бізнес вимог. Тобто наша система повинна обчислювати запит за певний проміжок часу, на що буде впливати кількість оперативної пам'яті, яку буде використовувати наше програмне забезпечення. В якості прикладу розглянемо запити до баз даних, що використовують великі розрахунки та багато значень з бази. Їх краще виконувати на серверній стороні, а не на стороні клієнта, щоб не передавати користувацькій стороні обчислюємі данні коли достатньо відправити лиш результат.
- До нефункціональних вимог треба віднести зовнішній інтерфейс, бізнес правила, атрибути якості, та певні обмеження.

На даному етапі необхідно провести роботи з виявлення, аналізу, тестування, узгодження вимог

Висновки

В досліджені були представлені різні моделі розробки програмного забезпечення. Розглянуто переваги і недоліки кожного з них окремо і зведено до табличного вигляду. Для створення додатку системи обліку було запропоновано комбінувати каскадну і ітеративну модель. Такий метод дозволить розробити робочу версію продукту малою командою, визначить всі вимоги для продукту, встановить чіткі терміни розробки, та надає можливість протестувати продукт на кожному окремому модулі. Схематично відображено комбіновану модель для розробки додатку обліку майна в збройних силах України. В запропонованій моделі вказано послідовність етапів життєвого циклу та взаємодія між ними. Важливо що в комбінованій моделі життєвого циклу ми можемо з кожного етапу перейти до попереднього, але всі етапи мають бути відпрацьовані.

Література

1. Орловський С., Стаценко Д. Дослідження можливостей автоматизації системи ведення обліку в Збройних силах України [Електронний ресурс]. – Режим доступу: https://heraldts.khmnu.edu.ua/index.php/heraldts/article/view/926/1339.
2. Популярні життєві цикли розробки ПЗ [Електронний ресурс]. – Режим доступу: https://training.qatestlab.com/blog/technical-articles/popular-software-development-life-cycles/.
3. Берко А. Ю., Верес О. М. Організація баз даних: практичний курс : навч. посіб. для студ. / Нац. ун-т «Львів. політехніка». – Львів, 2003. – 149 с.
4. Енциклопедія кібернетики : у 2 т. / за ред. В. М. Глушкова. – Київ : Гол. ред. Української радянської енциклопедії, 1973.
5. Навчальний посібник з дисципліни «Технології розробки програмного забезпечення» для студентів спеціальності 123 «Комп'ютерна інженерія».
6. Коберн А. Сучасні методи опису функціональних вимог до систем.

References

1. Orlovskyi S., Statsenko D. Doslidzhennia mozhlyvostei avtomatyzatsii systemy vedennia obliku v Zbroinykh sylakh Ukrainy [Elektronnyi resurs]. – Rezhym dostupu: https://heraldts.khmnu.edu.ua/index.php/heraldts/article/view/926/1339.
2. Populiarni zhyttievi tsykly rozrobky PZ [Elektronnyi resurs]. – Rezhym dostupu: https://training.qatestlab.com/blog/technical-articles/popular-software-development-life-cycles/.
3. Berko A. Yu., Veres O. M. Orhanizatsiia baz danykh: praktychnyi kurs : navch. posib. dlia stud. / Nats. un-t «Lviv. politekhnika». – Lviv, 2003. – 149 c.
4. Entsyklopediia kibernetyky : u 2 t. / za red. V. M. Hlushkova. – Kyiv : Hol. red. Ukrainskoi radianskoi entsyklopedii, 1973.
5. Navchalnyi posibnyk z dystsypliny «Tekhnolohii rozrobky prohramnoho zabezpechennia» dlia studentiv spetsialnosti 123 «Kompiuterna inzheneriia».
6. Kobern A. Suchasni metody opysu funktsionalnykh vymoh do system.