МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Методичні рекомендації до виконання комплексного курсового проекту

для студентів спеціальності "Інформаційні управляючі системи та технології" всіх форм навчання

Харків. Вид. ХНЕУ, 2010

Затверджено на засіданні кафедри інформаційних систем.
Протокол № 6 від 17.03.2010 р.

М54 Методичні рекомендації до виконання комплексного курсового проекту для студентів спеціальності "Інформаційні управляючі системи та технології" всіх форм навчання / укл. І. О. Золотарьова, С. В. Мінухін, Ю. Е. Парфьонов, Р. К. Бутова, Т. О. Свердло. – Харків : Вид. ХНЕУ, 2010. – 80 с. (Укр. мов.)

Наведено методичні рекомендації щодо розробки комплексного курсового проекту, зміст і рекомендації до виконання окремих розділів та підрозділів, оформлення проекту.

Рекомендовано для студентів спеціальності "Інформаційні управляючі системи та технології".

Вступ

Методичні рекомендації призначені для студентів, що навчаються за спеціальністю "Інформаційні управляючі системи і технології", і містять методичні та організаційні питання розробки комплексного курсового проекту, який охоплює комплекс навчальних дисциплін природничо-наукового спрямування:

"Програмна інженерія";

"Організація баз даних та знань";

"CASE-технології";

"Об'єктно-орієнтоване програмування"

та професійного спрямування:

"Сучасні засоби програмування";

"Web-програмування";

"Інформаційні системи в сучасному бізнесі".

Мета даних методичних рекомендацій – надати методичну допомогу студентам у дослідженні і аналізі питань з розроблення проектних рішень стосовно створення та організації роботи автома-тизованого модуля в АІС конкретного об'єкта управління відповідно до вимог стандартів на інформаційні системи і технології, вироблених світовою практикою.

Методичні рекомендації ознайомлять студентів з тематикою комплексного курсового проектування, з вимогами, пропонованими до комплексного курсового проекту, до його змісту й оформлення, з організацією роботи над проектом.

1. Завдання курсового проектування

Курсове проектування є завершальним етапом вивчення циклу дисциплін природничо-наукового та професійного спрямування.

Робота над комплексним курсовим проектом сприяє систематизації, поглибленню й закріпленню знань, отриманих студентами при вивченні навчальних дисциплін: "Програмна інженерія", "Організація баз даних та знань", "CASE-технології", "Об'єктно-орієнтоване програмування", "Сучасні засоби програмування", "Web-програмування", "Інформаційні системи в сучасному бізнесі".

У процесі курсового проектування студентами будуть розвинуті навички практичного застосування отриманих знань й здобуті компетенції з аналізу, моделювання й розроблення модуля АІС конкретного об'єкта з використанням CASE-інструментів, сучасних технічних та інстру-

ментальних засобів розроблення АІС. При цьому студент повинен показати вміння користуватися спеціальною літературою, довідниками, ДСТ з інформаційних технологій, інтернет-виданнями.

У розробленому комплексному курсовому проекті студент повинен показати, що він володіє компетенціями з:

аналізу предметної області, структурних і функціональних особливостей об'єкта управління;

аналізу і виявлення існуючих бізнес-проблем з управління об'єктом;

виділення у структурі АІС об'єкта управління конкретного функціонального модуля;

моделювання і аналізу бізнес-процесів модуля з використанням сучасних CASE-інструментів;

виділення комплексу інформаційно зв'язаних задач, автоматизація рішення яких спрямована на вирішення бізнес-проблеми, що виявлена на етапі аналізу;

аналізу інтерфейсу і функціональності готових програмних продуктів, які автоматизують рішення комплексу задач модуля, з метою використання рішень, реалізованих у програмному продукті, як прототипу при розробленні проектних, технічних рішень та документації комплексного курсового проекту;

розроблення специфікацій бізнес-вимог до модуля, що розробляється;

розроблення функціональних вимог до комплексу задач модуля з використанням CASE-інструменту Rational Rose;

розроблення проектного документа "Опис постановки комплексу задач";

розроблення математичної постановки вирішення комплексу задач;

опису інформаційних потоків з використанням CASE-інструментів BP Win або Rational Rose;

проектування структури бази даних з використанням CASE-інструментів ER Win або Rational Rose;

написання програмного коду для вирішення комплексу задач модуля з використанням сучасних інструментальних засобів та мов програмування високого рівня;

розроблення інтерфейсу програмного продукту;

тестування розробленого програмного продукту;

перевірки працездатності програмного продукту шляхом машинного вирішення комплексу задач на даних контрольного приклада.

Робота над комплексним курсовим проектом певною мірою визначає загальнотеоретичну спеціальну підготовку студента і готує його до

виконання більш складного й завершального етапу навчального процесу – дипломного проектування. Студент повинен розглядати роботу над комплексним курсовим проектом як "генеральну репетицію" дипломного проектування.

2. Організація курсового проектування

Відповідно до навчального плану розроблення і захист комплексного курсового проекту студенти денної форми навчання виконують у 9 семестрі, заочної форми навчання – у 11 семестрі.

Керівництво курсовим проектуванням здійснюється викладачами кафедри інформаційних систем, які є одночасно майбутніми керівниками дипломних проектів тих студентів, які розробляють комплексні курсові проекти.

Якісне виконання комплексного курсового проекту вимагає чіткої організації роботи студентів з моменту вибору теми проекту й до його захисту. Студенту надається право вільного вибору теми проекту з урахуванням його схильності та можливостей найбільш повно застосовувати отримані знання.

Тематика комплексних курсових проектів відповідає сучасним напрямкам розвитку й удосконалення діючих АІС й орієнтована на дослідження та розроблення питань створення модулів АІС різних предметних областей: промислових підприємств, банківських установ, страхових компаній, фінансово-кредитних установ, фірм різних напрямків підприємницької діяльності.

При розробленні модулів АІС необхідно орієнтуватися на використання технологій електронних комунікацій, інтернет-технологій, нобільних технологій, які дозволяють створити нові комп'ютеризовані бізнес-процеси.

Такі рішення сприяють підвищенню ступеня автоматизації бізнес-процесів й бізнес-функцій, рівня комунікацій. Крім того, необхідно орієнтуватися на інформаційні технології, інструменти і методи, які забезпечують створення процесів бізнес-аналітики нового рівня. Це технології і інструменти Сховищ Даних, OLAP (Online Analytical Processing).

Рекомендована тематика комплексного курсового проекту наведена в додатку А. Можливе розширення запропонованих тем у межах навчальних дисциплін навчального плану за фахом у зв'язку з постійним розвитком сучасних інформаційних технологій.

Запропонована студентом тема комплексного курсового проекту повинна бути погоджена з керівником проекту.

Підготовка студентів денної форми навчання до курсового проектування починається під час проходження ними у 8-му семестрі проектно-технологічної практики. Матеріали, зібрані за місцем проходження проектно-технологічної практики, можуть бути покладені в основу розробки комплексного курсового проекту. Бажано, щоб комплексний курсовий проект був розроблений з орієнтацією на базу проходження переддипломної практики. Також бажано, щоб проектні рішення комплексного курсового проекту були покладені в основу розроблення рішень з дипломного проекту. Студенти заочної форми навчання можуть виконувати комплексні курсові проекти за матеріалами, зібраними за місцем їхньої роботи.

Для затвердження обраної теми комплексного курсового проекту студент подає заяву на ім'я завідувача кафедри інформаційних систем. Після затвердження обраної теми на засіданні кафедри студенту видається завдання на комплексне курсове проектування (додаток Б).

У завданні наводиться тема комплексного курсового проекту, вхідні дані до нього, зміст пояснювальної записки, завдання на розроблення автоматизованого модуля, на розроблення програмного продукту (додатка), строки початку і закінчення роботи над комплексним курсовим проектом, обумовлені графіком навчального процесу.

Керівник проекту орієнтує студента на розроблення проектних рішень із використанням сучасних інструментальних засобів та інформаційних технологій, використовуючи процесний підхід і сучасні архітектурні рішення до розроблення інформаційних систем.

Вхідною базою для розроблення проектних рішень є матеріали конкретного об'єкта управління, зібрані під час проектно-технологічної практики, і функціональні характеристики проаналізованих програмних продуктів, обраних як прототип.

Розроблені проектні рішення повинні відповідати сучасному рівню розвитку інформаційних технологій, комп'ютерної техніки, засобів електронних комунікацій, інструментальних засобів.

Студент розробляє зміст комплексного курсового проекту, обговорює його з керівником, розробляє анотацію до комплексного курсового проекту (додаток В), підготовляє вхідні дані та приступає до проектування. У процесі проектування студент повинен регулярно відвідувати консультації керівника, надавати на перевірку йому робочі матеріали відповідно до затвердженого графіка курсового проектування. Комплексний курсовий проект студент повинен виконувати самостійно.

Проект, оформлений відповідно до викладених вимог, студент здає на перевірку керівникові за тиждень до строку захисту.

Захист комплексних курсових проектів організовується кафедрою інформаційних систем за два тижні до екзаменаційної сесії згідно з графіком, затвердженим завідувачем кафедри.

Захист здійснюється із демонстрацією роботи на ПК розробленого додатка на даних контрольного прикладу й презентацією розроблених проектних рішень.

3. Структура, зміст і обсяг комплексного курсового проекту

Комплексний курсовий проект складається з пояснювальної записки й графічного матеріалу, підготовленого у вигляді презентаційного матеріалу, який демонструється при захисті проекту. До проекту додається машинний носій із записаним розробленим програмним продуктом.

Обсяг пояснювальної записки становить не більше 40 сторінок надрукованого на ПК тексту. Таблиці, діаграми, відеокадри, машинограми, вхідні документи, лістинг роздрукованого програмного коду виносяться в додатки.

Основні розділи пояснювальної записки проекту наведені у табл. 1.

Таблиця 1

Основні розділи пояснювальної записки проекту та їхні обсяги

Назва розділу	Кількість сторінок
Титульний аркуш	1
Завдання на комплексне курсове проектування	1
Реферат	1
Зміст	2
Вступ	2
Розділ 1. Постановка завдань дослідження та проектування модуля <назва модуля>	12
Розділ 2. Проектні й технічні рішення та документація	20
Висновки	1
Використана література	
Додатки	

Графічний матеріал, підготовлений у вигляді презентації, повинен включати:

моделі аналізу предметної області у вигляді:

CASE-діаграм;

UML-діаграм;

схеми бази даних;

звіти – вихідні машинограми з показниками, які розраховані на даних контрольного прикладу, що отримані в результаті роботи розробленого бізнес-додатка (прикладної програми);

демонстраційний ролик, який показує роботу бізнес-додатка.

Рекомендується така структура пояснювальної записки комплексного курсового проекту:

РЕФЕРАТ

ЗМІСТ

ВСТУП

РОЗДІЛ 1. ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ ТА ПРОЕКТУВАННЯ МОДУЛЯ <назва модуля>

1.1. Змістовний опис і аналіз предметної області, структурних і функціональних особливостей об’єкта управління

1.2. Огляд і аналіз існуючих варіантів розв’язання комплексу задач модуля <назва модуля>

1.3. Розроблення специфікацій бізнес-вимог до модуля <назва модуля>

1.4. Висновки й пропозиції

РОЗДІЛ 2. ПРОЕКТНІ Й ТЕХНІЧНІ РІШЕННЯ ТА ДОКУМЕНТАЦІЯ

2.1. Деталізація вимог до модуля <назва модуля>

2.1.1. Глосарій проекту

2.1.2. Розроблення функціональних вимог

2.1.3. Опис постановки комплексу задач модуля <назва модуля>

2.1.4. Математична постановка комплексу задач

2.1.5. Опис інформаційних потоків

2.1.6. Проектування структури бази даних

2.2. Розроблення програмного продукту

2.2.1. Розроблення інтерфейсу програмного продукту

2.2.2. Склад та взаємодія програмних модулів

2.2.3. Результати тестування програмного продукту

2.2.4. Контрольний приклад

ВИСНОВКИ

Використана література

Додатки

4. Методичні рекомендації до розроблення розділів і підрозділів комплексного курсового проекту

Реферат – це короткий виклад змісту пояснювальної записки, що включає основні фактичні відомості та висновки, необхідні для початкового ознайомлення з пояснювальною запискою до комплексного курсового проекту. Реферат має бути розміщений після задачі на комплексне курсове проектування.

Реферат містить:

дані про обсяг пояснювальної записки, кількість ілюстрацій, таблиць, додатків, джерел у переліку бібліографічних посилань;

текст, який відображує інформацію, надану у пояснювальній записці, в такій послідовності:

об'єкт дослідження;

мета проекту;

методи дослідження й розробки;

результати та їх новизна;

основні технологічні характеристики та показники;

рекомендації щодо використання результатів комплексного курсового проекту;

область застосування;

висновок та прогноз стосовно розвитку розробок проекту;

перелік ключових слів.

Реферат необхідно складати обсягом не більше 500 слів.

Ключові слова існують для розкриття суті проекту та для розповсюдження інформації про розробку. Розміщують їх після тексту реферату. Перелік містить 5 – 15 ключових слів (словосполучень), написаних у рядок прописними літерами в називному відмінку, через кому.

Приклад реферату наведено у додатку Д.

Зміст пояснювальної записки включає:

вступ;

послідовно перераховані найменування всіх розділів, підрозділів, пунктів і підпунктів (якщо вони мають заголовки);

висновки;

використану літературу;

найменування додатків.

У змісті вказуються номери сторінок, на яких міститься початок матеріалу розділу, підрозділу, пункту, підпункту.

У **вступі** необхідно ідентифікувати та сформулювати бізнес-проблему, яка виникла на підприємстві (вказати назву підприємства) у веденні бізнесу і ділових операцій, обґрунтувати актуальність розроблюваної теми проекту для вирішення цієї бізнес-проблеми на основі автоматизації розв'язання комплексу задач проектованого модуля. Коротко охарактеризувати функціональність модуля, яку бажано досягти при автоматизації комплексу задач, які є об'єктом розробки комплексного курсового проекту. Необхідно охарактеризувати технічну та програмну платформу розробки автоматизованого модуля.

Необхідно сформулювати мету та задачі комплексного курсового проекту.

Мета проекту – це проектування, розроблення та реалізація автоматизації процесів автоматизованого вирішення комплексу задач на базі сучасних інструментальних та технічних засобів <найменування засобів>.

Досягнення мети проекту здійснюється шляхом вирішення в процесі проектування таких задач:

виділення предметної області управління бізнесом, яка підлягає автоматизації;

детальне моделювання виділеної предметної області з визначенням етапів, підетапів, бізнес-процесів, бізнес-функцій, процедур;

аналіз інтерфейсу та функціональності готового програмного продукту, призначеного для автоматизації виділених бізнес-процесів;

розроблення специфікацій бізнес-вимог до модуля, що розробляється;

розроблення документу "Опис постановки комплексу задач модуля";

розроблення програмного продукту для автоматизації вирішення комплексу задач.

Розділ 1. Постановка завдань дослідження та проектування модуля <назва модуля> є аналітичним.

Метою розділу 1 є проведення детального аналізу бізнес-проблеми, яка виникла на об'єкті управління (підприємстві) при веденні бізнесу й ділових операцій, та вибір шляхів її вирішення.

У результаті проведеного аналізу повинні бути виявлені недоліки, "вузькі місця" в існуючій інформаційній системі, розроблені заходи з її удосконалення для того, щоб проектований модуль ІС відповідав характеристикам системи-прототипу та вимогам управління бізнесом підприємства.

У підрозділі 1.1. Змістовний опис і аналіз предметної області, структурних і функціональних особливостей об'єкта управління необхідно:

коротко описати напрямки бізнес-діяльності об'єкта управління; подати схему організаційної структури управління; виявити, яка існує бізнес-проблема в розвитку бізнесу підприємства та його конкурентоспроможності на ринку.

Такими бізнес-проблемами можуть бути:

зниження конкурентоспроможності продукції підприємства на ринку;

низька лояльність клієнтів до підприємства, до його продукції;

відсутність стабільної клієнтської бази;

недостатня оперативність процесів управління;

паперовий документообіг;

відсутність сучасних інструментів для проведення аналізу господдарської та фінансової діяльності;

недостатня якість процесів управління та ін.

Тут необхідно виділити ті бізнес-процеси та бізнес-функції, які пов'язані з вирішенням даної бізнес-проблеми, вказати, які підрозділи організаційної структури їх виконують.

Для цього необхідно виконати моделювання бізнес-процесів з використанням CASE-інструментів.

Моделювання та аналіз бізнес-процесів дозволяють студентові глибоко розібратися у предметній області проектованого модуля, щоб нова система не автоматизувала неефективні або застарілі бізнес-процеси.

У процесі моделювання необхідно виділити, як мінімум, **три бізнес-задачі** в модулі, які повинні бути інформаційно зв’язані та реалізовувати єдину технологію процесу управління об'єктом.

У складі задач модуля одна з них повинна бути аналітичною.

У процесі моделювання необхідно виділити транзакційну складову бізнес-процесу, яка забезпечує збір, накопичення та обробку кількісних даних про поточний стан об’єкта управління, а також аналітичну складову, яка забезпечує аналіз кількісних показників, сформованих у транзакційні складові.

Аналітична складова бізнес-процесу повинна забезпечити дослідження кількісних показників у різних розрізах та вимірах: за періодами часу, за товарами (продукцією), за клієнтами, підрозділами.

Проведення такого багатоаспектного аналізу забезпечить інформаційну підтримку прийняття бізнес-рішень, спрямованих на вирішення виявленої бізнес-проблеми.

У процесі виділення бізнес-задач у модулі необхідно правильно сформулювати назву кожної задачі. При цьому дотримуватися таких правил. У назві задачі повинна бути присутня назва бізнес-функції управління або бізнес-процесу, які підлягають автоматизації при автоматизованому вирішенні задачі. Тобто має бути названий об'єкт автоматизації. Крім того, в назві задачі повинен бути присутній період, за який або на який розраховуються формовані в задачі техніко-економічні показники бізнес-діяльності. При цьому прийменник **"за"** використо-вується при формулюванні назви задачі облікового, контрольного або аналітичного характеру. Наприклад, задача "Облік продажів продукції за місяць", або "Аналіз клієнтської бази підприємства за квартал", або "Контроль виконання графіка відвантаження продукції покупцям за місяць".

Прийменник "на" використовується при формулюванні назви задачі планового, прогнозного характеру. Наприклад, "Формування календарного плану постачань продукції покупцям на місяць" або "Прогнозування показників фінансової діяльності підприємства на планований рік".

При використанні інтернет-технологій і мобільних технологій для автоматизованого вирішення задач назва задачі може формулюватися так: "Управління процесом формування та оброблення замовлень клієнтів на основі Web-технологій", або "Автоматизація процесів продажів продукції з використанням мобільних технологій", або "Управління процесами співпраці з бізнес-партнерами підприємства на основі Web-технологій".

У результаті моделювання повинна бути створена ієрархія діаграм, яка відображує за рівнями декомпозиції задач виділені бізнес-функції та механізми, за допомогою яких ці функції виконуються, а також дані й інформацію на вході і виході, які зв'язують між собою виконувані бізнес-функції (роботи).

Цей комплекс діаграм є функціональною моделлю предметної області модуля "Як є" (As-Is).

Ця модель використовується для аналізу та удосконалення існуючих бізнес-процесів при їх автоматизації шляхом виявлення і вилучення елементів, які повторюються, та "паперових" технологій їхнього виконання.

Даний підрозділ повинен закінчуватися висновками про те, які бізнес-процеси та їх елементи повинні бути автоматизовані.

Підрозділ 1.2. Огляд і аналіз існуючих варіантів розв'язання комплексу задач модуля <назва модуля> повинен містити аналіз функціональності й інтерфейсу одного або двох програмних продуктів, які призначені для автоматизації бізнес-процесів розроблюваного модуля.

Метою даного підрозділу є вивчення кращого практичного досвіду провідних фірм-розробників програмних продуктів та використання їх рішень і методології при розробленні проектних рішень комплексного курсового проекту. Це буде сприяти тому, що розроблюваний у проекті програмний продукт буде відповідати сучасним потребам ринку програмних продуктів для бізнесу.

Необхідно розкрити такі питання: призначення програмного продукту: фірма-розробник; склад програмних модулів та їх характеристики; привести інтерфейсні вікна; вихідні документи (звіти), які формуються; охарактеризувати переваги від використання програмного продукту.

За допомогою CASE-інструментів необхідно створити удосконалену функціональну модель автоматизованих бізнес-процесів у рамках даного додатка та виконати її аналіз. Це модель "Як повинно бути" (To-Be).

На початкових етапах створення ІС необхідно побудувати модель за допомогою графічної мови IDEF0.

IDEF0 – методологія (сукупність прийомів) функціонального моделювання і графічного описання процесів, яка призначена для формалізації і опису бізнес-процесів. Особливістю IDEF0 є її акцент на ієрархічне представлення об'єктів, що значно полегшує розуміння предметної області. В IDEF0 розглядаються логічні зв'язки між роботами, а не послідовність їх виконання у часі, а також відображаються всі сигнали управління.

IDEF0 будується на таких концептуальних засадах:

1. Система та модель.

Модель – штучний об’єкт, що відображає компоненти системи. Система – сукупність сутностей (об'єктів) та зв’язків між ними, які виокремленні із середовища на певний час і з певною метою, тобто це сукупність міцно пов’язаних об’єктів, які мають властивості організації, зв’язності, цілісності та розчленованості.

Модель відображає, що відбувається в системі та як нею керують, які "входи" вона перетворює та які засоби використовуються для виконання цих функцій.

2. Блочне моделювання та його графічне представлення.

Основний концептуальний принцип методології IDEF0 – представлення будь-якої системи у вигляді набору взаємопов'язаних блоків, що відображають процеси, операції, дії, які відбуваються у системі.

3. Строгість та формалізм.

Розробка моделей IDEF0 потребує дотримання формальних правил, що забезпечує перевагу методології відносно точності та цілісності складних багаторівневих моделей.

4. Ітеративність моделювання.

Розробка моделей IDEF0 це ітеративна процедура. На кожному з етапів ітерації розробник пропонує власний варіант моделі, який обговорюється, рецензується та цикл повторюється. Подібна організація роботи дозволяє оптимально використовувати знання системного аналітика, що володіє методологією IDEF0 та знання фахівців.

5. Відокремлення "організації" від "функції".

При розробці моделей слід уникати прив'язування функції досліджуваної системи до існуючої організаційної структури, це допомагає нівелювати суб’єктивну точку зору керівництва та організації в цілому. Організаційна структура має бути результатом аналізу моделі. Порівняння результатів з існуючою структурою дозволить оцінити адекватність моделі, та запропонувати рішення спрямовані на вдосконалення цієї структури.

В основі графічної мови IDEF0 лежить чотири базових поняття:

функціональний блок (робота);

інтерфейсні дуги;

декомпозиція;

глосарій.

Функціональний блок (робота) – поіменовані процеси, задачі чи функції, які виконуються протягом визначеного відрізку часу та мають конкретні результати. Назва роботи має бути сформульована віддієслівним іменником ("Виготовлення деталі", "Переробка сировини", "Аналіз продажів", "Формування заказу", "Підбір путівок").

Відповідно до принципів побудови діаграм за технологією IDEF0, кожній з функцій (робіт) відповідає певний блок. На IDEF0-діаграмі, головному документі при аналізі та проектуванні системи, блок є прямокутником (рис. 1). Кожна із чотирьох сторін прямокутника має конкретне призначення:

– верхня сторона має значення "управління" (control);

– ліва сторона має значення "входу" (input);

– права сторона має значення "виходу" (output);

– нижня сторона має значення "механізм".

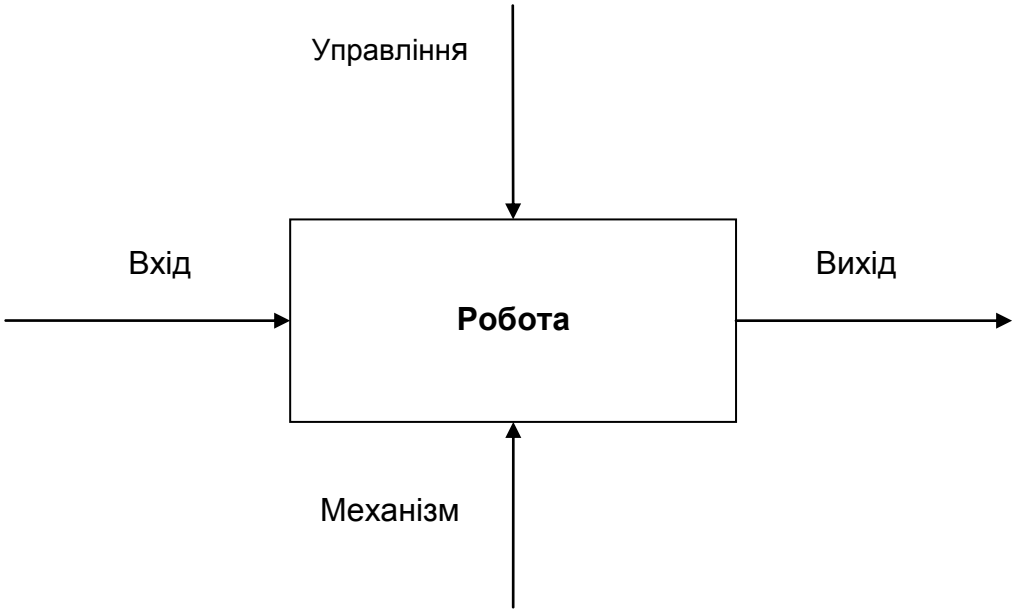

Рис. 1. **Блок діаграми IDEF0**

Через інтерфейсну дугу відбувається взаємодія з іншими блоками чи із зовнішнім середовищем, відображається стрілкою. Вона відображає елемент системи, який оброблюється функціональним блоком або впливає на функцію, що відображена цим блоком. Залежно від того, з якої сторони підходить інтерфейсна дуга, вона має назву "вхідної", "вихідної", "управляючої" чи "механізм".

Отже, взаємодія системи з навколишнім світом описується як вхід (дещо, що перероблюється системою), вихід (результат діяльності системи), управління (стратегії та процедури, під управлінням яких виконується робота) та механізм (ресурси, необхідні для виконання роботи). Тобто, система, що перебуває під управлінням, перетворює вхід на вихід, використовуючи при цьому механізми.

Відповідно до стандарту IDEF0, кожен функціональний блок повинен мати не менш ніж одну управляючу та вихідну інтерфейсну дугу.

Не залежно від виду, кожна дуга має бути поіменована іменником чи іменниковим словосполученням ("Специфікація", "Звіт з продажів", "Договір на поставку", "Стратегія рекламної акції", "Процесор", "Деталь №111", "Цінні папери", "Аналітик", "Провідний спеціаліст").

Принцип декомпозиції використовується для розділення складного процесу на його складові. При цьому рівень деталізації процесу визначається безпосередньо розробником моделі. Декомпозиція дозволяє поступово структурувати модель системи у вигляді ієрархічної структури окремих діаграм, що робить її менш перевантаженою та легшою для розуміння.

Модель IDEF0 завжди починається з представлення системи як цілого – одного функціонального блоку з інтерфейсними дугами, що виходять за межі моделі. Така діаграма має назву контекстна діаграма та має ідентифікатор "A-0". Функціональний блок на такій діаграмі має назву комплексу вирішуваних задач у курсовому проекті.

Під час декомпозиції, функціональний блок, який у контекстній діаграмі відображає систему як ціле, зазнає деталізації на іншій діаграмі. Діаграма другого рівня містить зміст робіт, що відбивають головні підфункції роботи контекстної діаграми та є дочірніми по відношенню до неї. У свою чергу, функціональний блок контекстної діаграми має назву батьківського по відношенню до дочірніх. Кожна підфункція дочірньої діаграми може зазнати подальшої деталізації шляхом аналогічної декомпозиції. Суттєвим є те, що в кожному випадку декомпозиції функціонального блоку всі інтерфейсні дуги фіксуються на дочірній діаграмі.

При декомпозиції системи всі стрілки, що входять у неї та виходять з неї, переходять на діаграму декомпозиції. Їх необхідно зв'язати з роботами, що зображують підсистеми.

На діаграмах декомпозиції для зв'язку підсистем між собою використовуються внутрішні стрілки, що виходять з однієї роботи і входять в іншу.

Оскільки IDEF0-діаграми містять у собі складну та концентровану інформацію, для того щоб обмежити перевантаженість діаграм, у стандарті введено такі правила:

обмеження кількості функціональних блоків (робіт) на діаграмі від трьох до шести;

обмеження кількості підходящих до одного функціонального блоку інтерфейсних дуг (не більше чотирьох);

кожна робота повинна мати бодай одну управляючу інтерфейсну дугу й одну вихідну;

робота може мати нуль та більше вхідних інтерфейсних дуг та дуг-механізмів;

назви функціональних блоків (робіт) та інтерфейсних дуг не можуть містити лише одне з таких слів: функція, діяльність, процес, вхід, вихід, управління чи механізм. Робота повинна мати назву сформульовану віддієслівним іменником ("Аналіз продажів", "Формування наказу на переведення", "Підбір турів"). Назва інтерфейсних дуг формулюється іменником чи іменниковим словосполученням ("Специфікація", "Звіт з продажів", "Договір на поставку", "Стратегія рекламної акції", "Цінні папери", "Аналітик", "Провідний спеціаліст").

Підрозділ 1.3. Розроблення специфікацій бізнес-вимог до модуля <назва модуля>. Усі вимоги повинні бути сформульовані з точки зору потреб бізнесу, тобто кожна вимога повинна відповідати певній бізнес-меті.

Якість розроблення вимог сприяє розробленню додатка з припустимим рівнем якості.

Розроблені специфікації вимог повинні включати три рівня вимог:

по-перше, бізнес-вимоги, які описують бізнес-мету, яку необхідно досягти підприємству з точки зору розвитку бізнесу. Наприклад, сформувати базу лояльних клієнтів, або збільшити обсяги продажів продукції до певного рівня. Це ринкові вимоги, які є "статутом" проекту. Бізнес-вимоги становлять каркас усього проекту, але не надають розробнику достатньої інформації для створення продукту. Вони визначають, наскільки продукт відповідає тенденціям ринку, розвитку інформаційних технологій, корпоративній стратегії розвитку підприємства;

по-друге, вимоги користувача, того, хто безпосередньо буде взаємодіяти з продуктом і буде виконувати свої функції автоматизовано. Тут значну увагу приділяють якісним характеристикам розроблюваного продукту;

по-третє, функціональні вимоги, які визначають функціональність, яку надає додаток, щоб користувачі змогли досягти бізнес-мети або розв'язати існуючу бізнес-проблему. Практично це специфікація функцій, яка передається на розроблення проекту і програмного продукту.

На основі вимог визначаються також особливості тестування програмного продукту і документації для користувачів.

В підрозділі за допомогою CASE-інструментів створюється діаграма бізнес-варіантів використання.

Метою розробки моделі бізнес-процесів (в Rational Rose – Business Use Case Model) є встановлення бізнес-процесів, що підлягають автоматизації, зв'язків між ними і цілей, які вони підтримують.

Модель бізнес-процесів, які підлягають автоматизації, слід використовувати під час декомпозиції системи на підсистеми на етапі визначення вимог до системи.

Кожному з виділених бізнес-процесів пропонується у подальшому на етапі визначення вимог до системи поставити у відповідність підсистему в системі, що проектується.

Якщо бізнес-процеси, що розглядаються для автоматизації, незалежні один від одного, то у системі, яка проектується, підсистеми також будуть функціонально незалежні одна від одної.

На основі цілей бізнес-процесів, що підлягають автоматизації, повинні бути сформульовані цілі розроблюваної системи.

Для зображення власне бізнес-процесу повинен використовуватися елемент бізнес-процес (в Rational Rose – Business Use Case).

Бізнес-процеси повинні називатися віддієслівними іменниками ("Відвантаження готової продукції", "Формування звіту продаж", "Підбір турів").

Для відображення зв'язків між бізнес-процесами використовують зв'язки залежності з такими стереотипами:

включає – "include";

розширює – "extend";

узагальнює ("батько-нащадок") – "generalization".

Зв'язок "залежність" між бізнес-процесами зі стереотипом "включає" ("include"), використовується, коли різні бізнес-процеси містять у собі один і той же бізнес-процес. Для зв'язку зі стереотипом "включає" стрілку слід направити від бізнес-процесу, що містить у собі інший бізнес-процес, до того який включається. Зв'язок відображається пунктирною лінією з назвою стереотипу.

Деякі бізнес-процеси можуть виконуватися при настанні певних умов або бути опціональними. У цьому випадку слід використовувати зв'язок-залежність зі стереотипом "розширює" ("extend"). Цей зв'язок зручно використовувати при відображенні бізнес-процесів, які повинні виконуватися у виняткових ситуаціях або при настанні певних умов. Для зв'язку зі стереотипом "розширює" стрілку слід направляти від бізнес-процесу, який розширює, до бізнес-процесу, який розширюється іншим бізнес-процесом. Зв'язок відображається пунктирною лінією з назвою стереотипу.

Зв'язок зі стереотипом "узагальнює" ("батько-нащадок", "generalization") повинен використовуватися, коли необхідно щоб бізнес-процес нащадок володів усіма властивостями бізнес-процесу батька та можливо якимись додатковими властивостями. Зв'язок зі стереотипом "узагальнює" відображається суцільною лінією з великою трикутною стрілкою. Стрілка направляється від "нащадка" до "батька".

Для зображення суб'єкта або об'єкта, який є ініціатором бізнес-процесу або споживачем результатів бізнес-процесу, необхідно використовувати елемент "бізнес-роль".

Підрозділ 1.4. Висновки й пропозиції повинен містити висновки про необхідність створення автоматизованого модуля з певною функціональністю, яка відповідає аналізованому програмному продукту (вказати назву програмного продукту). Пояснити, чому даний програмний продукт

обраний як прототип для розроблення проекту. Запропонувати, які інструментальні засоби будуть використовуватися для розроблення автоматизованого модуля.

Розділ 2. Проектні й технічні рішення та документація призначений для проектування комплексу задач модуля з використанням сучасних інформаційних технологій та розроблення працездатного бізнес-додатка з використанням розвинутих інструментальних і технічних засобів. Тут необхідно чітко визначити основні категорії користувачів бізнес-додатків, його функціональність, уяснити образ і межі проекту.

Підрозділ 2.1. Деталізація вимог до модуля <назва модуля> повинен містити специфікації вимог до проектованого модуля, тобто необхідно виявити ті властивості, якими повинен володіти готовий програмний продукт, щоб становити якусь цінність для користувачів.

Пункт 2.1.1. Глосарій проекту повинен містити три підпункти:

2.1.1.1. Словник термінів, понять, категорій предметної області модуля.

2.1.1.2. Основні категорії користувачів.

2.1.1.3. Опис найбільш важливих документів предметної області.

У підпункті *2.1.1.1.* необхідно скласти словник термінів, понять, категорій як предметної області, так і проектних рішень, які необхідно знати користувачеві для розуміння специфікації вимог до програмного продукту. Обов'язково привести пояснення цих термінів, понять, категорій.

У підпункті *2.1.1.2.* необхідно визначити основні категорії користувачів майбутньої системи та описати можливості, які вони одержать від впровадження комплексу задач модуля. Наприклад, "користувач зможе оформити замовлення через Інтернет", або "користувач зможе отримати автоматизовано звіт за місяць, в якому…". Можливості необхідно описувати на загальному рівні, доступному для розуміння усіма користувачами. Тобто, описати, **що** користувач зможе виконати, а **не як** він це буде реалізовувати.

До основних категорій користувачів відносяться:

кінцевий користувач системи, який буде виконувати свої функції за допомогою програмного продукту, буде вводити вхідну інформацію, запускати комплекс задач на вирішення, контролювати хід вирішення задач, отримувати результати розрахунків, аналізувати їх, приймати рішення, передавати отримані результати у вигляді машинограм або в електронному вигляді іншим користувачам;

інші користувачі, які отримають результати розрахунків та використовують їх для прийняття рішень з управління бізнес-діяльністю. Але вони безпосередньо не працюють з програмним продуктом.

У *підпункті 2.1.1.3.* необхідно описати найбільш важливі документи предметної області (вхідні і вихідні) та їх призначення. Це можуть бути як паперові документи, так і електронні у вигляді первинних документів або сформованих звітів.

Пункт 2.1.2. Розроблення функціональних вимог призначений для визначення, в першу чергу, основних діючих осіб, котрі беруть участь у роботі системи, і створення діаграми варіантів використання, що відображає функціональність, яка буде реалізована в програмному продукті, що розроблюється. Необхідно навести опис потоків подій і діаграми послідовності.

Дійові особи (актори) – це активні компоненти системи, які беруть участь у її роботі. Тут необхідно визначити ролі різних дійових осіб, у залежності від їх бізнес-функцій, обов'язків, розмежування доступу до БД і функцій системи, способів використання.

Варіант використання у діаграмі розглядається як інструмент підвищення інформативності функціональних вимог. У діаграмі повинна бути наведена кожна функція, яка реалізується системою, має конкретний закінчений результат, корисний користувачеві.

Метою розроблення сукупності діаграм варіантів використання є відображення меж системи, що розробляється. Варіант використання – це певна функціональність системи, що розробляється, її складова частина, сервіс, що система надає користувачеві.

Деякі ділові підзадачі можуть бути автоматизовані (виконуватися актором за допомогою системи, що розробляється) або виконуватися автоматично системою без участі актора-людини.

Для кожного бізнес-варіанта, що автоматизується, слід розглянути необхідний перелік сервісів системи для підтримки автоматизованого його вирішення.

Актор (actor) – узгоджена множина ролей, які грають зовнішні сутності по відношенню до варіантів використання при взаємодії з ними.

Актор представляє будь-яку зовнішню по відношенню до системи, що моделюється, сутність, яка взаємодіє з системою і використовує її функціональні можливості для досягнення певних цілей або вирішення окремих задач. Кожен актор може розглядатися як якась окрема роль щодо конкретного варіанта використання. Стандартним графічним позначенням актора на діаграмах є зображення "чоловічка", під якою записується ім'я актора.

Ім'я актора має бути достатньо інформативним з точки зору семантики. Для цієї мети підходять найменування посад у компанії (наприклад, продавець, касир, менеджер, президент). Не рекомендується давати акторам імена власні або назви моделей конкретних пристроїв.

Актори використовуються для моделювання зовнішніх по відношенню до проектованої системи сутностей, які взаємодіють з системою. У ролі акторів можуть виступати інші системи, у тому числі активні елементи проектованої системи або її окремі класи. Важливо розуміти, що кожен актор визначає узгоджену множину ролей, у яких можуть виступати користувачі цієї системи в процесі взаємодії з нею. У кожний момент часу з системою взаємодіє цілком певний користувач, при цьому він виступає в одній з таких ролей. Найбільш наочний приклад актора – конкретний відвідувач web-сайта в Інтернет зі своїми параметрами аутентифікації.

Оскільки в загальному випадку актор завжди знаходиться поза системою, його внутрішня структура ніяк не визначається. Для актора має значення тільки його зовнішнє уявлення, тобто те, як він сприймається з боку системи. Актори взаємодіють з системою за допомогою передачі і прийому повідомлень від варіантів використання. Повідомлення – це запит актором сервісу від системи та отримання цього сервісу. Ця взаємодія може бути виражене за допомогою асоціацій між окремими акторами і варіантами використання. Крім цього, з акторами можуть бути пов'язані інтерфейси, які визначають, яким чином інші елементи моделі взаємодіють з цими акторами.

Опису особливостей взаємодій між варіантами використання та інтерфейсами використовують такі ж види зв’язків, як і під час опису бізнес-процесів.

Кожен варіант використання повинен мати опис.

Діаграми взаємодії (interaction diagrams) описують поведінку груп об'єктів, що взаємодіють. Зазвичай діаграма взаємодії охоплює поведінку об'єктів у рамках тільки одного варіанта використання. На такій діаграмі відображається ряд об'єктів і ті повідомлення, якими вони обмінюються між собою.

Розрізнюють такі види повідомлень:

інформаційне повідомлення (informative) – повідомлення, що забезпечує об'єкт-одержувач інформацією для оновлення його стану;

повідомлення-запит (interrogative) – повідомлення, що запитує видачу інформації про об'єкт-одержувача;

імперативне повідомлення (imperative) – повідомлення, що запитує у об'єкта-одержувача виконання дій.

Існують два види діаграм взаємодії: діаграми послідовності (sequence diagrams) і кооперативні діаграми (collaboration diagrams).

Діаграми послідовності відображають потік подій, що відбувається в рамках варіанта використання. Один варіант використання може мати декілька можливих послідовностей. Зазвичай під сценарієм розуміється конкретний екземпляр потоку подій.

На діаграмі послідовності об'єкт зображується у вигляді прямокутника на вершині пунктирної вертикальної лінії. Ця вертикальна лінія називається лінією життя (lifeline) об'єкта. Вона є фрагментом життєвого циклу об'єкта у процесі взаємодії. Кожен об’єкт діаграми є екземпляром того чи іншого класу.

Кожному класу моделі необхідно дати унікальне ім'я. Зазвичай імена класів не містять пропусків. Також слід екземпляру класа призначити стереотип класу – механізм, що дозволяє розділяти класи на категорії.

У мові UML основними стереотипами є: граничні (Boundary), сутність (Entity) і управління (Control).

Граничні класи (boundary classes)  – це класи, які розташовані на межі системи і навколишнього середовища. Вони включають усі форми, звіти, інтерфейси з апаратурою (такі, як принтери або сканери) та інтерфейси з іншими системами. Для того, щоб знайти граничні класи, треба дослідити діаграми варіантів використання. Кожній взаємодії між дійовою особою і варіантом використання повинен відповідати принаймні один граничний клас. Саме такий клас дозволяє дійовій особі взаємодіяти із системою.

Не обов'язково створювати унікальні граничні класи для кожної пари "дійова особа – варіант використання". Якщо дві дійові особи ініціюють один і той же варіант використання, вони можуть застосовувати загальний граничний клас для взаємодії із системою.

Класи-сутності (entity classes)  відображають основні поняття (абстракції) предметної області і, як правило, містять інформацію, що зберігається постійно. Як правило, клас-сутність характеризується наявністю таблиці в базі даних, що його відображає (або запису). Вимоги визначають потік подій. Потік подій визначає об'єкти, класи та атрибути класів. Кожен атрибут класу-сутності стає полем у базі даних. Застосовуючи такий підхід, легко відстежувати відповідність між полями бази даних і вимогами до системи, що зменшує вірогідність збору непотрібної інформації.

Управляючі класи (control classes)  відповідають за кординацію дій інших класів. Зазвичай у кожного варіанта використання є один управляючий клас, який контролює послідовність подій цього варіанта використання. Управляючий клас відповідає за координацію, але сам не несе в собі ніякої функціональності – решта класів не посилає йому великої кількості повідомлень. Натомість він сам посилає множину повідомлень. Управляючий клас просто делегує відповідальність іншим класам, з цієї причини його часто називають класом-менеджером. У системі можуть застосовуватися і інші управляючі класи, загальні для декількох варіантів використання.

Крім згаданих вище стереотипів, можна створювати і власні.

Кожне повідомлення зображується у вигляді стрілки між лініями життя двох об'єктів. Повідомлення відображаються у тому порядку, як вони виникають у потоці подій. Кожне повідомлення позначається номером та ім'ям повідомлення; також можна додати аргументи і деяку управляючу інформацію і, крім того, показати самоделегування (self-delegation) – повідомлення, яке об'єкт посилає самому собі, при цьому стрілка повідомлення вказує на ту ж саму лінію життя.

Після виконання розміщення об’єктів та повідомлень на діаграмі необхідно надати певних властивостей встановленим повідомленням.

Simple – проста посилка повідомлення;

Synchronous – операція відбувається тільки в тому випадку, коли клієнт посилає повідомлення, а сервер може прийняти повідомлення клієнта;

Balking – операція відбувається тільки в тому випадку, коли сервер готовий негайно прийняти повідомлення. Якщо сервер не готовий до прийому, клієнт не видає повідомлення;

Timeout – клієнт відмовляється від видачі повідомлення, якщо сервер протягом певного часу не може його прийняти;

Asynchronous – клієнт видає повідомлення, і, не чекаючи відповіді серверу, продовжує виконання свого програмного коду.

Не всі об’єкти, які показані на діаграмі послідовності, явно присутні в потоці подій. Там може не бути форм для заповнення, але їх необхідно показати на діаграмі послідовності, щоб дозволити дійовій особі ввести нову інформацію в систему або проглянути її. У потоці подій швидше за

все також не буде й управляючих об’єктів (control objects). Ці об’єкти управляють послідовністю подій у варіанті використання.

Подібно до діаграм послідовності, кооперативні діаграми від ображають потік подій через конкретний сценарій варіанта використання. Діаграми послідовності упорядковані за часом, а кооперативні діаграми загострюють увагу на зв'язках між об'єктами.

Діаграми кооперацій містять всю ту інформацію, що і діаграми послідовності, але кооперативна діаграма по-іншому описує потік подій. З неї легше зрозуміти зв'язки між об'єктами, проте важче з'ясувати послідовність подій. З цієї причини часто для деяких сценаріїв створюють діаграми обох типів.

На кооперативній діаграмі, так само як і на діаграмі послідовності, стрілками позначають повідомлення, обмін якими здійснюється у рамках даного варіанта використання. Їх часова послідовність вказується шляхом нумерації повідомлень.

Діаграми послідовності і кооперацій слід розмістити у додатках до курсового проекту.

Пункт 2.1.3. Опис постановки комплексу задач модуля <назва модуля> припускає розроблення сукупності проектних рішень, які є базою для розроблення інформаційного забезпечення комплексу задач, програмного коду їх вирішення та технологічного забезпечення.

В основу розроблення документа "Опис постановки комплексу задач" повинні бути покладені дані опису предметної області модуля і розроблені вище специфікації бізнес-вимог та функціональних вимог. Документ "Опис постановки комплексу задач модуля" містить три підпункти:

2.1.3.1. Характеристика комплексу задач.

2.1.3.2. Вихідна інформація.

3.1.3.3. Вхідна інформація.

Підпункт 2.1.3.1. "Характеристика комплексу задач" включає 7 параграфів:

1) призначення комплексу задач;

2) перелік об’єктів, при управлінні якими розв’язується комплекс задач;

3) періодичність і тривалість вирішення комплексу задач;

4) умови, при яких припиняється вирішення комплексу задач автоматизованим способом;

5) зв'язки комплексу задач з іншими задачами АІС;

6) посади осіб та назви підрозділів, які визначають умови і тимчасові характеристики вирішення комплексу задач;

7) розподіл дій між персоналом і технічними засобами при різних ситуаціях вирішення комплексу задач.

Зміст параграфу 1 "Призначення комплексу задач" такий:

склад комплексу задач модуля, їх кодові позначення, складовою частиною якого бізнес-процесу, якої функціональної підсистеми АІС є комплекс задач, на АРМі якого управлінського працівника розв’язується;

мета вирішення комплексу задач;

призначення комплексу задач;

образ проекту автоматизованого вирішення комплексу задач;

доцільність автоматизованого вирішення комплексу задач.

Рекомендації до розроблення параграфу 1 "Призначення комплексу задач"

Склад комплексу задач модуля повинен включати бізнес-задачі, які були виділені на етапах моделювання предметної області і розроблення бізнес-вимог.

Кодове позначення задач, що вирішуються в АІС, включає 4 знаки. Структура кодового позначення має такий вигляд (рис. 2)

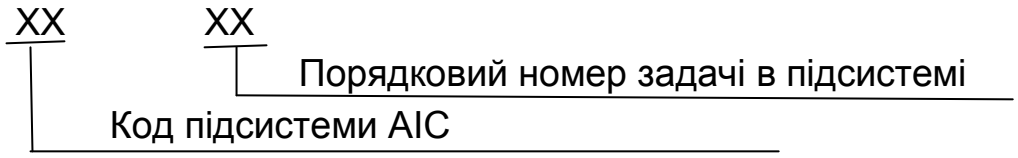

Рис. 1. **Структура кодового позначення задачі в АІС**

Тобто старші два розряди коду позначають функціональну підсистему АІС, до якої відноситься проектована задача, молодші два розряди – порядковий номер задачі в підсистемі.

Наприклад, в АІС промислового підприємства підсистема "Управління збутом і реалізацією готової продукції" позначена кодом 06, а задача "Ведення реєстру договорів на постачання продукції", яка відноситься до даної підсистеми, має код 0601.

У АБС підсистема "Управління кредитними ресурсами" позначається кодом 02, а задача "Формування портфеля заявок на кредити", що відноситься до даної підсистеми, має код 0201.

На наведених прикладах видно, що код задачі залежить від того, на якому об'єкті (підприємстві, установі) функціонує АІС, а предметна область визначає її функціональну структуру, тобто розбиття АІС на підсистеми.

При цьому задача може бути складовою певного бізнес-процесу і модуля, що автоматизуються в АІС. Необхідно вказати *місце і роль*, що відводиться комплексу задач у названих бізнес-процесах і модулі.

Технологія вирішення задач у сучасних АІС ґрунтується на децентралізованій обробці інформації, яка характеризується тим, що персональні комп'ютери (ПК) встановлюються в місцях виникнення і споживання інформації – на робочих місцях управлінського персоналу. На базі ПК організовується автоматизоване робоче місце (АРМ) конкретного управлінського працівника, який виступає в ролі кінцевого користувача АІС. Кінцевий користувач бере активну участь в реалізації автоматизованого інформаційного процесу управління шляхом вирішення комплексу задач на своєму АРМі. АРМи в АІС мають проблемно-професійну орієнтацію на конкретну предметну область, тобто на ті функції, які виконує кінцевий користувач з управління об'єктом, працюючи в його конкретному підрозділі.

Основне призначення АРМа – автоматизоване вирішення задач і виконання розрахунків для надання кінцевому користувачу інформації, необхідної для виконання покладених на нього функцій та ухвалення ефективних управлінських рішень.

АРМ працює в комп’ютерній мережі об'єкта управління, обробка інформації здійснюється за технологією "клієнт-сервер". При цьому АРМ виконує функції "клієнта", тобто є робочою станцією користувача в комп’ютерній мережі об'єкта управління.

Виходячи з викладеного раніше, назва АРМа повинна формуватися за схемою: АРМ <назва посади кінцевого користувача> <назва підрозділу, в якому працює кінцевий користувач>. Наприклад, АРМ менеджера зі збуту відділу збуту, АРМ економіста планово-економічного відділу.

Ключовими моментами в характеристиці комплексу задач є визначення мети автоматизованого їх вирішення і призначення комплексу задач.

Мета вирішення комплексу задач – автоматизація конкретних функцій управління <прогнозування, планування, обліку, контролю, аналізу> або автоматизація управління бізнес-процесами <назва>.

Автоматизація функцій або бізнес-процесів впливає на підвищення якості, ефективності, оперативності системи управління бізнесом, що виражається в поліпшенні конкретних техніко-економічних показників бізнес-діяльності об'єкта.

Тому *призначення задачі – формування і розрахунок таких техніко-економічних показників:* <перелік техніко-економічних показників, що

розраховуються в задачі, приводиться в стовпчик після двокрапки>. Цей перелік показників міститься у формах вихідних документів, що формуються при вирішенні задач. Кожен показник розраховується за відповідною формулою, заданою алгоритмом вирішення задачі. При цьому необхідно приводити коректну і повну назву кожного показника. Наприклад, показник "Планований об'єм випуску продукції цехом на місяць, кількість, штук" або "Об'єм реалізації продукції за місяць, сума, тис. грн".

Опис автоматизованого варіанта вирішення комплексу задач переслідує мету розробити *образ проекту*, в якому показати переваги інформаційної технології (ІТ) їх вирішення, її відповідність сучасному рівню розвитку ІТ, охарактеризувати метод вирішення задачі, а також очікувані результати в управлінні бізнес-діяльністю від вирішення задач автоматизованим способом.

При цьому акцент робиться на використання засобів електронних телекомунікацій і технологію мережної обробки інформації.

Тут необхідно обґрунтувати інструментальні засоби розробки додатка, реалізовану додатком (програмним продуктом) функціональність, тобто ті функції, які реалізуються *автоматизовано*. Наприклад:

автоматизоване формування запиту <зміст запиту з вказівкою регламентований чи нерегламентований запит>;

автоматизоване формування первинного документа <назва документа> і його друк;

автоматизоване формування і друк звітів <назва звітів>;

генерація рішення (висновку) системи <назва рішення, висновку> для затвердження <назва посадовця, що затверджує рішення>.

При описі образу проекту за комплексом задач доцільно використовувати шаблони з ключових слів.

Проект, що розробляється, треба розглядати як бізнес-проект, тобто як проект спрямований на розвиток бізнесу, що заснований на використанні сучасних розвинутих інформаційних технологіях.

Спочатку треба описати бізнес-проблему, розв'язанню якої сприяє автоматизація вирішення комплексу задач. При описі треба використовувати ключові слова:

Бізнес-проблема <назва проблеми> торкається <точно викладати суть проблеми, в якому підрозділі виникла, хто з керівників повинен її вирішувати>, її наслідком є <викладати суть негативних наслідків>.

Потім вказати, від чого буде залежати успішне вирішення бізнес-проблеми і яка роль в цьому належить автоматизації бізнес-процесів і бізнес-функцій пропонованого комплексу задач.

Проектований модуль повинен стати *ефективним бізнес-рішенням*, заснованим на використанні сучасних бізнес-інструментів (методології) і тих IT-інструментів, які надаються користувачеві системи для вироблення і підтримки реалізації ефективних ділових рішень з управління бізнес-діяльністю. Необхідно вказати, які бізнес-інструменти та IT-інструменти надає система, які ділові результати та переваги вона забезпечує. Це можуть бути: OLAP-інструменти, інструменти графічного аналізу, інструменти постійного моніторингу.

Для розробки якісних IT-інструментів треба використати принципи та технології:

єдиного інформаційного простору системи;

корпоративного інформаційного порталу;

організації баз знань, сховищ даних;

OLAP-аналізу;

електронного документообігу;

інтернет, інтранет, екстранет;

мобільного зв’язку та ін.

Результатом використання цих технологій при розробці проекту і програмного продукту є оптимізовані комп'ютеризовані бізнес-процеси, <вказати які>, ефективні бізнес-рішення <розкрити, які бізнес-рішення приймаються, чому вони ефективні>.

Потім необхідно охаратеризувати користувальницьке середовище, тобто цільову аудиторію (кінцевого користувача і споживачів результатної інформації вирішення комплексу задач) і які можливості вони отримають від впровадження комплексу задач. Наприклад: доступ до інформації <вказати якої>; відстеження процесу <розкрити суть процесу>; реалізувати в реальному часі нерегламентовані запити до системи <описати зміст запитів>; реалізувати спільну роботу <вказати з ким>; підвищити інформованість про <вказати які процеси> і т. д.

Необхідно вказати, які переваги забезпечить автоматизація вирішення комплексу задач порівняно з ручною технологією його вирішення, наскільки інформаційна технологія його вирішення відповідає рівню розвитку сучасних інформаційних технологій, яку функціональність реалізує програма вирішення комплексу задач.

Відповідність рівню розвитку сучасних інформаційних технологій відображає програмно-технічна платформа вирішення комплексу задач,

тобто які програмні засоби (ОС, СУБД, мови програмування) і моделі ПК і серверів покладені в основу розробки автоматизованих процесів вирішення комплексу задач.

Доцільність автоматизованого вирішення комплексу задач обґрунтовується:

необхідністю поліпшення системи управління бізнес-процесами <назва бізнес-процесів> на основі створення інтегрованої інформаційної технології їх реалізації і впливом даного комплексу задач на поліпшення бізнес-процесів;

наявністю в базі даних необхідної інформації для вирішення комплексу задач <вказати якої інформації> і сформованої в результаті вирішення інших задач <вказати яких задач>;

великим об'ємом обчислювальних і логічних операцій з обробки інформації <обґрунтувати великі об'єми, наприклад, значною номенклатурою продукції, кількістю споживачів продукції>.

Зміст параграфу 2 "Перелік об'єктів, при управлінні якими розв'язується комплекс задач":

перелік технологічних об'єктів управління, підрозділів підприємства, для управління якими використовується результатна інформація, сформована в задачах (споживачі інформації);

перелік посадовців цих підрозділів, функції яких автоматизуються.

Рекомендації до розробки параграфу 2

Перерахувати, які функції кінцевого користувача автоматизуються, які інформаційні потреби задовольняються тих споживачів, яким передається результатна інформація, з вказівкою електронним способом чи кур'єром здійснюється передача.

Зміст параграфу 3 "Періодичність і тривалість вирішення комплексу задач":

календарні терміни формування і надання вихідної інформації;

періодичність вирішення задачі (за зміну, день, п'ятиденку, декаду, місяць, квартал, півріччя, рік);

обмеження за часом вирішення задачі (тривалість рішення).

Рекомендації до розробки параграфу 3

Календарні терміни формування вихідної інформації вказуються, якщо задача призначена для формування звітності. Тоді ці терміни узгоджуються з термінами надання звітності і з формою її передачі. Наприклад, щодня в електронному вигляді, у вигляді файла <ім'я файла> до дванад-

цятої години дня, наступного за звітним, на адресу <найменування одержувача>.

Для планових задач указується: "за 5 днів до початку планованого місяця", для облікових: "1-го числа місяця, наступного за звітним".

Періодичність вирішення задач узгоджується з періодом, вказаним в заголовку вихідних документів (машинограм). Наприклад, періодичність: щодня, щомісячно, якщо в заголовку машинограми вказано: за день, за місяць, або на місяць. Задачі інформаційно-довідкового обслуговування можуть розв'язуватися за запитом користувача, тоді, коли виникає потреба в отриманні певної інформації у зв’язку з виробничою або фінансовою ситуацією, що склалася.

Тривалість вирішення задач залежить від об'ємів інформації, що вводиться, оброблюваної, від продуктивності процесора і швидкості роботи програм вирішення задач. Тривалість указується в хвилинах.

Зміст параграфу 4 "Умови, при яких припиняється вирішення комплексу задач автоматизованим способом":

перелік умов, при яких комплекс задач не може бути вирішений автоматизованим способом.

Рекомендації до розробки параграфу 4

Необхідно вказати, через відсутність (своєчасного ненадходження) якої вхідної інформації не може бути вирішений комплекс задач, які порушення в достовірності та цілісності даних у базі даних приводять до неможливості вирішення задач та як це позначиться на вирішенні інших задач. Як причина може бути вказаний вихід з ладу ПК або каналу зв'язку, нез'єднання із сервером.

Зміст параграфу 5 "Зв'язки комплексу задач з іншими задачами АІС":

схема інформаційних зв'язків комплексу задач;

опис інформаційних зв'язків комплексу задач.

Рекомендації до розробки параграфу 5

У даному параграфі повинна бути розроблена схема інформаційних зв'язків комплексу задач. У схемі показуються джерела вхідної інформації на вході, а також на виході – споживачі результатної інформації.

Схема розробляється відповідно до вимог ГОСТ 19.701-90 "Схеми алгоритмів, програм, даних, і систем". У схемі використовуються умовні позначення (символи): даних, процесу, ліній і спеціальні символи.

У центрі схеми розміщується символ "Процес" з указівкою всередині символу назви модуля. На вході, зверху, показуються джерела інформації, які формують вхідну інформацію для вирішення задач. Назва джерела поміщається усередині символу. Це можуть бути назви: підрозділу підприємства, контрагента, іншої задачі АІС, іншого АРМа, суб'єктів зовнішнього середовища, а також єдина база даних.

На виході, внизу показуються одержувачі вихідної інформації. Назва одержувача поміщається усередині символу. Це можуть бути назви: підрозділів, посадовців, інших задач, інших АРМів.

При позначенні джерел і одержувачів інформації використовуються відповідні символи. Зв'язки позначаються лінією, якщо інформація передається без використання каналу зв'язку, і лінією-блискавкою, якщо інформація передається по каналу зв'язку. На лініях проставляються номери зв'язків, які описуються відразу під схемою. Якщо напрям лінії зверху – вниз або зліва – направо, то стрілка на лінії не проставляється. Якщо навпаки, знизу – уверх і справа – наліво, то стрілка ставиться.

Зміст параграфу 6 "Посади осіб і назви підрозділів, які визначають умови і тимчасові характеристики вирішення комплексу задач":

посада особи;

назва підрозділу, в якому працює цей посадовець.

Рекомендації до розробки параграфу 6

Указується начальник того підрозділу, в якому розв'язується комплекс задач.

Зміст параграфу 7 "Розподіл дій між персоналом і технічними засобами при різних ситуаціях вирішення комплексу задач":

функціональний розподіл дій між всіма категоріями користувачів, споживачів інформації і технічними засобами в процесі вирішення комплексу задач у діалоговому режимі з конкретизацією діалогу "Кінцевий користувач – ПК".

Рекомендації до розробки параграфу 7

Необхідно вказати, якою інструкцією користується при роботі з додатком кінцевий користувач, за які дії він відповідає відповідно до функцій режимів меню і підказками на екрані ПК, що йому необхідно зробити в тій або іншій ситуації при виконанні розрахунків.

Вказати функції ПК, сервера та каналів зв'язку.

Підпункт 2.1.3.2. "Вихідна інформація"

У підпункті "Вихідна інформація" необхідно описати склад і структуру вихідних повідомлень, сформованих при вирішенні комплексу задач

(машинограм, відеокадрів, вихідних масивів). Вихідні повідомлення, як результат вирішення задач, фактично відбивають цільове призначення задач – сформувати інформацію для прийняття управлінських рішень, тобто розрахувати значення конкретних показників бізнес-діяльності підприємства (прогнозних, планових, стратегічних, облікових, контрольних, аналітичних, маркетингових), оцінити показники і відбити їхню динаміку. Крім того, у вихідних документах можуть бути згенеровані системою готові управлінські рішення та рекомендації на основі розрахованих значень конкретних показників. У зв'язку з цим необхідно продумати склад вихідних повідомлень по кожній задачі і ретельно спроектувати їхню структуру і зміст з урахуванням інформативності та корисності кожного з повідомлень для цілей управління бізнес-діяльністю.

При цьому значну увагу приділити реалізації запитів кінцевого користувача до системи з метою отримання інформації, потреба в якій виникає у зв'язку з певною виробничою, господарською або фінансовою ситуацією.

Підпункт "Вихідна інформація" включає два параграфи: "Перелік і опис вихідних повідомлень" і "Перелік і опис структурних одиниць інформації вихідних повідомлень".

У параграфі "Перелік і опис вихідних повідомлень" необхідно навести їхній опис за формою табл. 2.

Таблиця 2

Перелік і опис вихідних повідомлень

Ідентифікатор (код повідомлення)	Найменування вихідного повідомлення	Форма представлення (МГ, ВК, масив)	Періодичність видачі	Термін видачі і припустимий час затримки вирішення	Одержувачі	Призначення вихідної інформації
1	2	3	4	5	6	7

Код повідомлення повинен містити 7 знаків відповідно до такої структури (рис. 3)

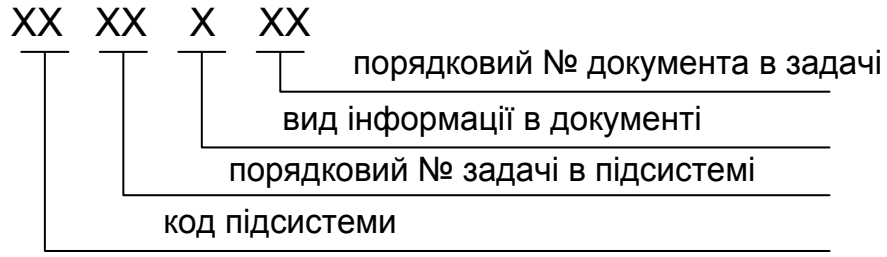

Рис. 3. **Структура кодового позначення повідомлення в АІС**

Вид інформації (п'ятий розряд) кодується так (табл. 3).

Таблиця 3

Кодування інформації

Інформація	Вхідна	Збережена	Вихідна
Нормативна	1	2	3
Планова, прогнозна	4	5	6
Облікова (контролю, аналізу)	7	8	9

Цим 7-значним кодом кодуються всі документи в АІС як вхідні, так і вихідні. Для вихідних масивів, сформованих у задачі, наводиться літерний ідентифікатор, а не цифровий код. У графі 2 вказується повне найменування машинограм (МГ), відеокадрів (ВК) або масивів.

Зміст граф 4 та 5 повинен бути узгоджений з параграфом "*Періодичність і тривалість вирішення комплексу задач*" характеристики комплексу задач, а зміст графи 6 – зі схемою інформаційних зв'язків (на виході).

Для вихідних масивів у графі 5 написати: "разом із МГ <код МГ>".

Для масивів у графі 6 наводиться перелік кодів задач, які використовують сформовані вихідні масиви як вхідні. Зміст цієї графи повинен бути узгоджений зі схемою інформаційних зв'язків задачі. Ці коди задач наведені у схемі на виході задачі.

У графі 7 "Призначення вихідної інформації" необхідно сформулювати, які управлінські рішення можуть ухвалюватися на основі значень показників вихідних повідомлень – МГ та ВК.

Для вихідних масивів у графі 7 написати: "є вхідною інформацією для рішення задачі (задач)".

У параграфі "Перелік і опис структурних одиниць інформації вихідних повідомлень" необхідно описати показники, реквізити, сигнали управління, що мають самостійне значеннєве значення, за формою табл. 4.

Таблиця 4

Перелік і опис структурних одиниць вихідних повідомлень

Найменування структурної одиниці інформації	Ідентифікатор	Вимоги до точності
1	2	3

Опис структурних одиниць у табл. 4 необхідно привести за кожним вихідним повідомленням, у тому числі описати реквізити, які розміщуються у заголовку МГ або ВК.

Підпункт 2.1.3.3. "Вхідна інформація"

У підпункті "Вхідна інформація" потрібно навести перелік і опис вхідних документів (за формою табл. 5) і вхідних масивів для вирішення задач (за формою табл. 6).

Таблиця 5

Перелік і опис вхідних документів

Ідентифікатор документа	Найменування документа	Термін надходження	Частота надходження	Постачальник документа
1	2	3	4	5

Якщо при вирішенні задач автоматизовано формується і друкується вхідний первинний документ, тобто проводиться автоматизоване документування господарської або фінансової операції, то цей документ описується в табл. 5. Наприклад, в задачі 0605 автоматизовано, за допомогою ПК формується і друкується первинний документ "Накладна на відвантаження продукції зі складу покупцю". Тоді в графі 3 написати: "у міру оформлення здійснюваних господарських (фінансових) операцій по <найменування операції>", в графі 4: "щодня", а в графі 5 – "формується автоматизовано в задачі <код вирішуваного завдання>".

Таблиця 6

Характеристика масивів вхідної інформації

Найменування масиву	Ідентифікатор (ім’я) масиву	Тип масиву	Технологія формування
1	2	3	4

Типи масивів можуть бути: НДІ, оперативні, з іншої задачі <найменування задачі>.

Технологія формування: в єдиній БД; на іншому АРМі <найменування АРМа>; в іншій задачі <найменування задачі>; у багато-віконному режимі на підставі інформації первинного документа <ідентифікатор документа із табл. 5> і масивів НДІ <імена масивів>; у процесі вирішення задачі (наприклад, масив залишків може бути вхідним і вихідним).

Перелік і опис структурних одиниць інформації вхідних повідомлень привести за формою табл. 7.

Таблиця 7

Перелік і опис структурних одиниць вхідних документів

Найменування структурної одиниці	Точність числового значення	Джерела інформації (документ)	Ідентифікатор джерела інформації
1	2	3	4

У *пункті 2.1.4. "Математична постановка комплексу задач"* потрібно обґрунтувати вибір економіко-математичної моделі вирішення кожної задачі, якщо задача є оптимізаційною або задачею прогнозування. Економіко-математична модель включає цільову функцію для обраного критерію ефективності та систему обмежень. Необхідно описати послідовність розрахунку результатних показників відповідно до логіки реалізації економіко-математичної моделі.

Якщо задача вирішується на основі алгоритму прямого лічення, необхідно навести розрахункові формули у вигляді математичної залежності вихідних показників, що розраховуються, від вхідних. При написанні формул повинні бути використані позначення (імена) показників і реквізитів, вказані в описі постановки комплексу задач.

Якщо задача не має математичного формулювання її вирішення, необхідно навести опис логіки послідовних дій у вигляді виконуваних функцій обробки інформації з задачі (добору, порівняння).

Математична і логічна моделі вирішення задачі повинні бути описані з достатнім ступенем деталізації, щоб за ними можна було написати програму розрахунку показників вихідних повідомлень.

Пункт 2.1.5. Опис потоків даних

Опис потоків даних у комплексі задач, що розглядається у курсовому проекті, слід виконувати за допомогою діаграм потоків даних (англ. *Data Flow Diagram*) – графічного представлення "потоків" даних у інформаційній системі. Ці діаграми також можуть бути використані для представлення обробки даних (структурна розробка).

Основними компонентами діаграм потоків даних є:

системи/підсистеми;

процеси;

потоки даних;

зовнішні сутності;

накопичувачі даних або сховища.

Інформаційна модель системи будується як деяка ієрархічна схема у вигляді контекстної діаграми, на якій початкова модель послідовно надається у вигляді моделі підсистем відповідних процесів перетворення даних. При цьому підсистема або система на контекстній діаграмі DFD зображується так само, як і процес – прямокутником із заокругленими вершинами.

Процес є сукупністю операцій з перетворення вхідних потоків даних у вихідні відповідно до певного алгоритму або правила. Як ім'я рекомендовано використовувати дієслово в невизначеній формі з не обхідними доповненнями.

Потік даних визначає якісний характер інформації, що передається через деяке з'єднання від джерела до приймача. Реальний потік даних може передаватися по мережі між двома комп'ютерами або будь-яким іншим способом, який припускає витягнення даних і їх відновлення в необхідному форматі. Потік даних на діаграмі DFD зображується лінією із стрілкою на одному з її кінців, при цьому стрілка вказує напрям потоку даних. Кожний потік даних має своє власне ім'я, що відображає його зміст.

Зовнішня сутність є матеріальним об'єктом або фізичною особою, які можуть виступати як джерело або приймач інформації. Визначення деякого об'єкта або системи як зовнішня сутність не є строго фіксованим. Хоча зовнішня сутність знаходиться за межами даної системи. Ім'я рекомендується використовувати як іменник у називному відмінку.

Накопичувач даних або сховище є абстрактним пристроєм або способом зберігання інформації, яка переміщується між процесами. Передбачається, що дані можна у будь-який момент помістити в накопичувач і через деякий час витягнути, причому фізичні способи переміщення і витягнення даних можуть бути довільними. Накопичувач даних може бути фізично реалізований різними способами, але найбільш часто передбачається його реалізація в електронному вигляді на магнітних носіях. При цьому як ім'я накопичувача рекомендується використовувати іменник, який характеризує спосіб зберігання відповідної інформації.

Таким чином, інформаційна модель системи в нотації DFD будується у вигляді діаграм потоків даних, які надаються графічно з використанням відповідної системи позначень.

У *пункті 2.1.6. Проектування структури бази даних (БД)* необхідно спроектувати CASE-засобами ER-Win або Rational Rose логічну та фізичну моделі бази даних модуля, що розробляється.

При проектуванні необхідно забезпечити такі властивості, які повинна мати БД, що розробляється:

функціональна повнота;
мінімальна надмірність;
цілісність БД;
цілісність домену;
цілісність таблиці;
цілісність посилання;
цілісність, обумовлена правилами бізнесу;
цілісність, пов'язана з поняттям несуперечливості даних;
узгодженість БД;
безпечність БД;
ефективність БД;
логічна і фізична незалежність;
розширюваність (відкритість) БД;
дружність інтерфейсу користувача.

Перераховані властивості БД розглядаються як вимоги до БД при її проектуванні.

Більшість вимог до БД пов'язані між собою. Вони задовольняються на різних етапах проектування БД різними засобами. Загальна вимога до всіх етапів проектування полягає в недопущенні дублювання даних.

Головним засобом забезпечення таких вимог до БД, як мінімальна надмірність, цілісність, несуперечливість, логічна та фізична не залежність, є нормалізація логічного подання даних.

Створена БД повинна бути спільною для багатьох додатків. Для цього необхідно використовувати сучасну промислову СУБД, яка підтримує позицію мережної обробки "клієнт-сервер" та забезпечує ефективний доступ користувачів до даних БД.

Підрозділ 2.2. Розроблення бізнес-додатка

Пункт 2.2.1. Розроблення інтерфейсу користувача

Інтерфейс користувача складається з елементів, які можуть перебувати в певних станах, а також у процесі взаємодії з користувачем програми переходити зі стану в стан. Він є системою, що управляється подіями (СУП).

Поводження таких систем найкраще характеризується їхньою реакцією на зовнішні події. Як правило, СУП перебуває в стані очікування, поки не одержить повідомлення про подію. Після того як СУП відреагує на подію, вона знову переходить у стан очікування наступної події. Для таких систем важливо визначити насамперед стійкі стани, події, що ініціюють переходи з одного стану в інший, і дії, що виконуються при зміні стану.

При формальному описі СУП доцільно використовувати UML-діаграми станів.

Діаграма станів зв'язує події й стани. При виникненні події наступний стан системи залежить як від її поточного стану, так і від події. Зміна стану називається переходом. Діаграма станів – це граф, вузли якого моделюють стани, а спрямовані дуги, позначені іменами відповідних подій, – переходи.

У даному пункті пояснювальної записки необхідно навести UML-діаграми станів, у яких можуть перебувати елементи інтерфейсу

користувача при виконанні кожного з бізнес-завдань модуля, а також їхній опис.

При розробленні інтерфейсу користувача необхідно керуватися методичними рекомендаціями, викладеними нижче.

Методичні рекомендації щодо розроблення інтерфейсу користувача

Користувальницький інтерфейс є своєрідним комунікаційним каналом, по якому здійснюється взаємодія користувача й комп'ютера.

Щоб створити ефективний інтерфейс, що здійснював би роботу із програмою приємною, потрібно розуміти, які завдання будуть вирішувати користувачі за допомогою даної програми і які вимоги до інтерфейсу можуть виникнути в користувачів.

Загальні принципи проектування інтерфейсу користувача:

1. Програма повинна допомагати виконувати завдання.

Це значить, що інтерфейс повинен бути легким для освоєння й не створювати перед користувачем перешкоду, яку він повинен буде подолати, щоб приступитися до роботи.

2. При роботі із програмою користувач не повинен відчувати дис.-комфорт.

Для реалізації цього принципу необхідно:

забезпечити перевірку результатів як можна більшого числа "некоректних" дій користувача, але не робити її повсюдно;

вказувати користувачеві, що саме йому робити, і виводити інформаційні повідомлення в ситуаціях, коли це дійсно необхідно;

надати досвідченим користувачам можливість відключення виведення інформаційних повідомлень;

добре продумувати зміст повідомлень, що виводяться користувачеві.

Широко відомі евристичні правила авторитетного американського фахівця в галузі проектування інтерфейсів Якоба Нільсена. Вони визначають мінімальні критерії, яким повинен відповідати інтерфейс будь-якої програми.

1. Наочність стану системи (правило зворотного зв'язку)

Система (у цьому випадку – комп'ютерна програма) повинна завжди інформувати користувача про стан своєї роботи за допомогою засобів зворотного зв'язку в прийнятний час.

При розгляді цього правила потрібно враховувати кілька аспектів:

користувач завжди повинен мати інформацію про поточний стан програми (наприклад, скільки часу пройшло від початку процесу копіювання файлів);

користувач обов'язково повинен бачити, до чого привело будь-яку його дію, наприклад, уведення даних, натискання кнопки;

вибір конкретного засобу зворотного зв'язку залежить від типу інформації, яку потрібно донести до користувача, а також типу дії, що викликає потребу в зворотному зв'язку.

Уважається, що якщо користувач виконав якусь дію й очікує результат його виконання, то цей результат (або повідомлення про помилку) повинен бути виведений в окремому діалоговому вікні. Якщо ж користувачеві необхідно надати поточну інформацію про процес, що не є прямим наслідком його дій, то можна обмежитися відображенням відповідного повідомлення в панелі стану.

Для організації зворотного зв'язку можуть бути використані й інші засоби. Найпопулярніші з них – звуки. Найбільш часто звукове оповіщення допомагає тоді, коли поява на екрані діалогових вікон є небажаною, а повідомлення в панелі стану можуть бути не помічені користувачем.

Важливо пам'ятати, що звукове оповіщення не повинне бути основним засобом організації зворотного зв'язку. Звук повинен лише доповнювати текстові повідомлення.

Проміжок часу, протягом якого користувач одержує інформацію про реакцію на його дію або про подію, повинен бути мінімальним. Це особливо важливо, тому що наявність або відсутність у користувача інформації про поточний стан системи визначає його подальші дії.

2. Відповідність між системою й реальним світом

Система повинна взаємодіяти з користувачем на його мові. У цьому випадку мається на увазі використання понять, образів і цілих концепцій, які знайомі користувачеві з реальної предметної області. У жодному разі не можна використовувати спеціалізовані терміни, які придатні тільки для професійних довідників з програмування.

Найпоширеніший приклад реалізації цього принципу – побудова користувальницького інтерфейсу, що імітує об'єкти реального світу. Наприклад, більшість із програм, що реалізують функції годинників, калькуляторів, програвачів компакт-дисків, записних книжок виглядають майже точно так, як їхні матеріальні аналоги. Знаменитий "сміттєвий кошик" на "робочому столі" операційної системи Windows, у яку можна "кинути" непотрібний файл або папку, – класичний приклад побудови інтерфейсу на основі об'єктів реального світу.

3. Управління користувачами та свобода їхніх дій

Користувачі повинні мати можливість управління системою й зміни її поточного стану. Однак, вони часто дають різні команди помилково (наприклад, випадково натиснувши кнопку або вибравши не той пункт меню). Отже, у користувача повинен бути "аварійний вихід" із цієї ситуації, чітко позначений у програмі. Найчастіше такий "вихід" реалізується у вигляді кнопки "Відміна", розташованої в діалоговому вікні, що дозволяє припинити виконання поточної операції або закрити це діалогове вікно. Крім цього, традиційним і тому звичним для більшості користувачів засобом "аварійного виходу" є натискання на клавіатурі клавіші <Escape>.

Хорошим тоном вважається сполучення цих способів "аварійного виходу".

Ще один засіб виходу з помилкової ситуації – команди "Undo" ("Відмінити") і "Redo" ("Повторити"). Вони є настільки зручними й підтримуються такою великою кількістю програм, що користувачі підсвідомо очікують, що будь-яку їхню дію можливо відмінити, повернувшись до попереднього стану.

Все це зобов'язує розроблювача дружнього користувальницького інтерфейсу комп'ютерної програми підтримувати дані команди. Якщо ж за якимись причинами дію, на виконання якої дав команду користувач, не можна відмінити, то на екран повинне буде виведене відповідне попередження, а також прохання підтвердити виконання команди.

4. Несуперечність і стандарти

Цей принцип означає використання тих самих засобів для вираження схожих образів і виконання дій, що мають однакову природу.

По-перше, це означає несуперечність при виборі засобів оповіщення про події та дії. Наприклад, інформація про поточний стан програми, звичайно виводиться в панелі стану, а повідомлення з результатами запитів користувача – в окремих діалогових вікнах.

Повідомлення про критичні помилки при цьому повинні сильно відрізнятися від звичайних інформаційних повідомлень: наприклад, вони можуть супроводжуватися різким звуком.

По-друге, це означає несуперечність при оформленні елементів користувальницького інтерфейсу. Наприклад, якщо він ґрунтується на класичному інтерфейсі Windows-додатків, що характеризується строгою колірною гамою, прямими лініями й кутами, то дуже дивним виглядало б рішення додати одному з вікон програми овальну форму й розфарбувати його яскравими кольорами.

По-третє, це означає несуперечність при виборі термінів. Користувачів не повинне спантеличувати те, що кілька різних понять, які використовуються в програмі, насправді означають те саме.

Головне – вирішити, що для позначення якоїсь конкретної дії або події буде застосовуватися один конкретний термін, що буде використовуватися певним способом (наприклад, слово "Інтернет" буде починатися із великої букви й не відмінюватися).

Принцип несуперечності – одне з найважливіших правил проектування користувальницьких інтерфейсів. Несуперечливий інтерфейс інтуїтивно зрозумілий і дуже легкий для освоєння, тому що при його вивченні користувач не зіштовхується із сюрпризами, і навіть ті частини інтерфейсу, які він бачить уперше, здаються йому давно знайомими.

5. Запобігання помилок

Стосовно теми проектування користувальницького інтерфейсу комп'ютерних програм, цей принцип означає таке: "Дизайн, що запобігає виникнення помилок, краще, ніж найкраще повідомлення про помилку".

6. Розуміння краще, ніж запам'ятовування

При розробленні інтерфейсу потрібно робити всі об'єкти, функції, дії легкодоступними користувачеві. Користувач не повинен постійно намагатися утримати в пам'яті інформацію з однієї частини програми, щоб застосувати її в іншій. У будь-який момент часу користувачеві повинно бути зрозуміло, що йому зараз потрібно робити. У хорошому інтерфейсі інструкції з використання системи доступні для виклику в будь-який час, коли це потрібно. Це може бути реалізоване як у вигляді продуманої організації елементів інтерфейсу, так і у вигляді підказок користувачеві.

Це правило відбиває принцип "прозорого" інтерфейсу – інтерфейсу, що зрозумілий і не змушує користувача згадувати, яку кнопку потрібно натиснути або який пункт меню вибрати в даний момент.

7. Гнучкість і ефективність використання

При проектуванні інтерфейсу користувача перед розроблювачем часто постає така проблема: потрібно, щоб інтерфейс був однаково зручний і для новачків, і для досвідчених користувачів.

Для рішення цієї проблеми використовують простий прийом: функції, які прискорюють роботу, оформлюють так, щоб їх не бачив початківець, але щоб вони були доступні досвідченим користувачам. Найпростіший приклад – це "гарячі клавіші", за допомогою яких можна швидко викликати функції програми, що часто виконуються. Позначення "гарячих клавіш" пишуться поруч із відповідними пунктами меню, тому вони, з одного боку, не заважають новачкам, а, з іншого боку, доступні досвідченим користувачам.

Інший приклад реалізації універсального користувальницького інтерфейсу – можливість виконати складні функції програми як за допомогою "майстра", що, немов за руку, "проведе" починаючого користувача по всіх етапах процесу, так і вручну, за допомогою настроювання опцій у відповідному діалоговому вікні.

8. Естетичний і мінімалістський дизайн

Це правило означає: "Нічого зайвого". Не потрібно захаращувати користувальницький інтерфейс програми елементами, які є недоречними й малокорисними. Справа в тому, що кожний елемент, будь то кнопка або текстовий підпис, обов'язково відволікає частину уваги користувача. Це може привести до того, що видимість і, відповідно, легкість сприйняття користувачем дійсно потрібних і корисних частин інтерфейсу буде сильно зменшена за рахунок елементів, без яких цілком можна було б обійтися.

9. Розпізнавання й виправлення помилок

"Допомагайте користувачеві розпізнавати й виправляти помилки" – говорить це правило.

Воно визначає проектування повідомлень про помилки. "Гарні" повідомлення про помилки – це повідомлення, які пояснюють, у чому заключається проблема й, найголовніше, як її виправити. Таким чином, "гарне" повідомлення про помилку повинне складатися із двох частин: опису помилки й опису вирішення проблеми.

Опис помилки повинен бути чітким, ясним і зрозумілим, давати користувачеві всю необхідну інформацію про причини й місце виникнення помилки. Найпростіше рішення – створити в довідковій системі

програми відповідний розділ, що роз'ясняє зміст проблеми й причини її виникнення. У самому ж діалоговому вікні з повідомленням про помилку може бути присутнім кнопка "Довідка" для виклику цього розділу.

Ще одним прикладом рішення даної проблеми є кнопка "Докладніше", при натисканні на яку діалогове вікно з повідомленням про помилку "розорюється", відображаючи більш докладну інформацію про причину виникнення збою.

Дуже важливо пам'ятати те, що повідомлення про помилку повинне містити її опис, а не числовий код помилки.

Існує багато програм, у яких повідомлення про помилки містять недостовірну інформацію, повідомляючи користувача зовсім не про ті проблеми, які виникли насправді. Тому при складанні описів помилок потрібно не забувати перевіряти коректність повідомлень, що генеруються програмою.

Відомо, що інформація про те, як виправити помилку або вирішити проблему має навіть більше значення, ніж опис помилки або проблеми. При описі шляху вирішення проблеми потрібно уникати складання занадто об'ємних текстів. В іншому разі користувачі будуть просто пробігати їх очами, не доходячи до змісту написаного. Найкраще скласти покрокову інструкцію, кожний крок якої складається з 1 – 2 речень.

10. Довідка й документація

Це правило полягає в необхідності надання користувачеві довідкової системи й документації до програми.

Проектування форм

Форми – це "будівельні блоки" інтерфейсу користувача.

Щоб створити добре спроектовану форму, необхідно усвідомити її призначення, спосіб і час використання, а також її зв'язки з іншими елементами програми.

Особливий вид форм – форми, призначені для введення даних. При їхній розробці основну увагу варто приділити швидкості їхнього використання. Основне правило – якщо користувач збирається ввести в базу даних велику кількість записів, то він не повинен підтверджувати введення кожної з них.

Щоб прискорити процес уведення даних, необхідно:

1. По можливості використовувати для додавання й редагування даних ту саму форму.
2. Призначати клавіатурні сполучення для команд.

3. Не змушувати користувача "перестрибувати" з однієї частини форми в іншу.

4. Не ставити процес уведення даних у залежність від вмісту окремих елементів управління форми.

5. Використовувати засоби зворотного зв'язку з користувачем.

Робота з декількома формами

Якщо додаток повинен містити кілька форм, то можна використовувати однодокументний (SDI) або багатодокументний (MDI) інтерфейс користувача.

Однак, у кожному разі взаємодія користувача з формами відбувається за допомогою обробки подій, що надходять від елементів управління форми. Тому, якщо в додатку передбачено кілька форм, то програму необхідно спроектувати так, щоб у користувачів не було можливості порушити послідовність її виконання (наприклад, вивести форму, для якої ще не готова інформація).

Ефективні меню

Ще одна важлива частина розробки форм – створення змістовних і ефективних меню. От деякі важливі рекомендації:

1. Додержуйтеся стандартних угод про розташування пунктів меню, прийнятим в операційній системі.

2. Групуйте пункти меню в логічному порядку й за змістом.

3. Для угруповання пунктів у меню, що розкриваються, використовуйте розділові лінії.

4. Уникайте надлишкових меню.

5. Уникайте пунктів меню верхнього рівня, які не мають меню, що розкриваються.

6. Не забувайте використовувати символ <…> для позначення пунктів меню, що активізують діалогові вікна.

7. Обов'язково використовуйте клавіатурні еквіваленти команд і "гарячі" клавіші.

8. Поміщайте на панель інструментів часто використовувані команди меню.

Нижче наведені деякі рекомендації із проектування Web-інтерфейсу користувача:

1. Мінімізуйте зусилля, які необхідно зробити користувачеві для прийняття рішень про навігацію.

2. По можливості використовуйте один екран. Використовуйте багатопанельні елементи управління, спливаючі вікна й майстри, щоб користувачі могли виконувати як можна більшу частину завдань, не використовуючи побічну навігацію.

3. При створенні декількох сторінок поєднуйте їх у розділи. Спростіть переміщення між розділами за допомогою основного елемента управління навігацією та додаткового елемента управління для переміщення між розділами.

4. Переконайтеся, що ключові елементи, такі як меню, заголовки й інша інформація, що відображається на всіх сторінках, оформлені одноманітно з візуальної точки зору.

5. Варто бути дуже обережним відносно навігаційних елементів управління для переміщення користувачів між внутрішніми сторінками різних розділів. Це можливо, але це може дезорієнтувати користувача.

6. Використовуйте додаткові засоби навігації, такі як дерева й карти вузлів, але не покладайтеся на них.

7. Завжди думайте про майбутнє розширення. Якщо в майбутньому очікується включення в продукт нових функцій, спочатку необхідно вирішити, як буде розширюватися навігація для ефективного включення нових екранів.

Пункт 2.2.2. Опис складу та взаємодії програмних модулів.

У цьому пункті пояснювальної записки має знаходитися:

1. UML-діаграма класів програми та її докладний опис.

2. UML-діаграми діяльності, що відповідають варіантам використання, та їх докладний опис.

3. Лістинг програми з коментарями (для класів – призначення класу; для методів – призначення методу, опис параметрів та значення, що повертається). Якщо програма реалізована з використанням програмної платформи .Net або Java, обов'язковим є використання документаційних XML- або JavaDoc-тегів відповідно.

Далі знаходяться методичні рекомендації щодо опис складу та взаємодії програмних модулів.

Найважливішими характеристиками архітектури будь-якої програмної системи є її структура й процес функціонування.

Під структурою програмної системи розуміють стійку в часі сукупність взаємозв'язків між її елементами. Залежно від рівня деталізації структурні елементи можуть бути підсистемами, процесами, бібліотеками й т. д.

Саме структура зв'язує воєдино всі елементи програмної системи й забезпечує її існування як єдиного цілого.

Процес функціонування програмної системи тісно пов'язаний зі зміною її властивостей або поводження в часі. Тому поряд з визначенням структурних елементів будь-яка архітектура визначає взаємодію між ними. Це такі взаємодії, які забезпечують бажане поводження системи.

Одним із принципів побудови моделей складних систем є принцип багатомодельності. Цей принцип є твердженням про те, що ніяка єдина модель не може з достатнім ступенем адекватності описувати різні аспекти складної системи.

Стосовно об'єктно-орієнтованого аналізу й проектування програмних систем це означає, що досить повна модель такої системи повинна містити деяке число взаємозалежних уявлень, кожне з яких адекватно відбиває деякий аспект її поводження або структури.

Найбільш загальними уявленнями складної системи прийнято вважати статичне й динамічне уявлення.

Для уявлення статичної структури моделі системи в термінології класів об’єктно-орієнтованого програмування призначена UML-діаграма класів. Вона може відбивати, зокрема, різні взаємозв'язки між окремими сутностями предметної області, такими як об'єкти й підсистеми, а також описує їхню внутрішню структуру й типи відносин.

Клас у мові UML служить для позначення безлічі об'єктів, які мають однакову структуру, поводженням і відносинами з об'єктами інших класів. Клас може не мати екземплярів або об'єктів. У цьому випадку він називається абстрактним класом. Графічно клас зображується у вигляді прямокутника, що додатково може бути розділений горизонтальними лініями на секції. У цих секціях можуть вказуватися ім'я класу, атрибути й операції.

Обов'язковим елементом позначення класу є його ім'я. Ім'я класу повинне бути унікальним у межах пакета, що описується деякою сукупністю діаграм класів. Воно вказується в першій верхній секції прямокутника. Рекомендується як імена класів використовувати іменники, записані без пробілів. Необхідно пам'ятати, що саме імена класів утворять словник предметної області. Прикладами імен класів можуть бути такі іменники, як "Співробітник", "Компанія", "Керівник", "Клієнт", "Продавець", "Менеджер", "Офіс" та, що мають безпосереднє відношення до

предметної області й функціонального призначення проектованої системи.

У другій зверху секції прямокутника класу записуються його атрибути. Кожному атрибуту класу відповідає окремий рядок тексту, що складається із квантора видимості атрибута, імені атрибута, його кратності, типу значень атрибута й, можливо, його вихідного значення.

Квантор видимості може приймати одне із трьох можливих значень:

1. Загальнодоступний (public). Атрибут із цією областю видимості доступний з будь-якого іншого класу пакета, у якому визначена діаграма.

2. Захищений (protected). Атрибут із цією областю видимості недоступний для всіх класів, за винятком підкласів даного класу.

3. Закритий (private). Атрибут із цією областю видимості не доступний для всіх інших класів без винятку.

Ім'я атрибута становить собою рядок тексту, що використовується як ідентифікатор відповідного атрибута й тому повинен бути унікальним у межах даного класу. Ім'я атрибута є єдиним обов'язковим елементом синтаксичного позначення атрибута.

Кратність атрибута характеризується загальною кількістю конкретних атрибутів даного типу, що входять до складу окремого класу.

Тип атрибута становить вираз, семантика якого визначається мовою специфікації відповідної моделі. У нотації UML тип атрибута іноді визначається залежно від мови програмування, яку передбачається використовувати для реалізації даної моделі. У найпростішому випадку тип атрибута вказується рядком тексту, що має осмислене значення в межах пакета або моделі, до яких ставиться розглянутий клас.

У третій зверху секції прямокутника записуються операції класу. Операція становить деякий сервіс, що надає будь-який екземпляр класу на певну вимогу. Сукупність операцій характеризує функціональний аспект поводження класу.

Кожній операції класу відповідає окремий рядок, що складається із квантора видимості операції, імені операції, вираження типу, що повертається операцією.

Ім'я операції становить рядок тексту, що використовується як ідентифікатор відповідної операції й тому повинен бути унікальним у межах даного класу.

Крім внутрішнього устрою або структури класів на відповідній діаграмі вказуються різні відносини між класами. При цьому сукупність типів

таких відносин фіксована в мові UML і визначена семантикою цих типів відносин. Базовими відносинами або зв'язками в мові UML є:

1. Відношення залежності.
2. Відношення асоціації.
3. Відношення узагальнення.
4. Відношення реалізації.

Відношення залежності в загальному випадку вказує деяке семантичне відношення між двома елементами моделі або двома безлічами таких елементів, що не є відношенням асоціації, узагальнення або реалізації. Відношення залежності використовується в такій ситуації, коли деяка зміна одного елемента моделі може потребувати зміни іншого залежного від нього елемента моделі.

Відношення асоціації відповідає наявності деякого відношення між класами. Найбільш простий випадок даного відношення – бінарна асоціація. Вона зв'язує в точності два класи й, як виключення, може зв'язувати клас із самим собою.

Тернарна асоціація й асоціації більш високого порядку в загальному випадку називаються N-арною асоціацією. Така асоціація зв'язує деяким відношенням три і більше класів, при цьому один клас може брати участь в асоціації більш ніж один раз.

Окремим випадком відношення асоціації є відношення агрегації, яке, в свою чергу, теж має спеціальну форму – відношення композиції.

Відношення агрегації має місце між декількома класами в тому випадку, якщо один із класів становить деяку сутність, що включає в себе як складові частини інші сутності. Дане відношення має фундаментальне значення для опису структури складних систем, оскільки застосовується для уявлення системних взаємозв'язків типу "частина-ціле". Розкриваючи внутрішню структуру системи, відношення агрегації показує, з яких компонентів складається система і як вони зв'язані між собою. Це відношення по своїй суті описує декомпозицію складної системи на більш прості складові частини, які також можуть бути піддані деком позиції, якщо в цьому виникне необхідність надалі.

Відношення композиції є окремим випадком відношення агрегації. Це відношення служить для виділення спеціальної форми відносини "частина-ціле", при якій складові частини в деякому змісті перебувають усередині цілого. Специфіка взаємозв'язку між ними полягає в тому, що

частини не можуть виступати у відриві від цілого, тобто зі знищенням цілого знищуються й всі його складові частини.

Відношення узагальнення є відношенням між більш загальним елементом (батьком або предком) і більш приватним або спеціальним елементом (дочірнім або нащадком). Стосовно до діаграми класів дане відношення описує ієрархічну будову класів і спадкування їхніх властивостей і поводження. При цьому передбачається, що клас-нащадок має всі властивості й поводження класу-предка, а також має свої власні властивості й поводження, які відсутні у класа-предка.

При моделюванні поводження проектованої або аналізованої системи виникає необхідність не тільки представити процес зміни її станів, але й деталізувати особливості алгоритмічної й логічної реалізації виконуваних системою операцій.

Для моделювання динамічних аспектів поводження системи в мові UML використовуються діаграми діяльності. Кожний стан на діаграмі діяльності відповідає виконанню деякої елементарної операції, а перехід у наступний стан спрацьовує тільки при завершенні цієї операції.

Графічно діаграма діяльності представляється у формі графа діяльності, вершинами якого є стани дії, а дугами – переходи від одного стану дії до іншого.

Стан дії є спеціальним випадком стану з деякою вхідною дією й принаймні одним переходом, що виходе зі стану. Звичайне використання стану дії полягає в моделюванні одного кроку виконання алгоритму (процедури) або потоку керування.

Простий перехід становить відношення між двома послідовними станами, що вказує на факт зміни одного стану іншим. Перебування об'єкта, що моделюється, в першому стані може супроводжуватися виконанням деяких дій, а перехід у другий стан буде можливий після завершення цих дій, а також після задоволення деяких додаткових умов. У цьому випадку говорять, що відбувається спрацьовування переходу. До спрацьовування переходу об'єкт перебуває в попередньому від нього стані, який називається вихідним станом, або в джерелі, а після його спрацьовування об'єкт перебуває у наступному стані (цільовому стані).

Перехід здійснюється при настанні деякої події: закінчення виконання діяльності, одержанні об'єктом повідомлення або прийомі сигналу. Спрацьовування переходу може залежати не тільки від настання деякої події, але й від виконання певної умови, яка називається сторожовою умо-

вою. Об'єкт перейде з одного стану в інше в тому випадку, якщо відбулася зазначена подія й сторожова умова прийняла значення "істина".

Діаграму діяльності можна визнати добре структурованою, якщо вона:

1. Сконцентрована на описі одного аспекту динаміки системи.
2. Містить тільки ті елементи, які істотні для розуміння цього аспекту.
3. Уявляє лише ті деталі, які відповідають даному рівню абстракції.

Пункт 2.2.3. Опис результатів тестування.

При розробці бізнес-додатка студент має провести модульне, інтеграційне та системне тестування.

У даному пункті пояснювальної записки повинен знаходитися опис процедур системного тестування та їхніх результатів.

Повинні бути описані такі види системного тестування:

1. Функціональне тестування.
2. Тестування безпеки.

Для опису процедури тестування повинні бути складені такі документи:

1. Тест-вимоги.
2. Тест-плани, сполучені зі звітами про проведення тестування.

У звіті про проведення тестування вказуються як позитивні так і негативні результати виконання окремих тестів.

Однак, загальний результат тестування додатка повинен бути позитивним, бо в протилежному випадку він не відповідає певним вимогам. Для цього в разі необхідності проводяться додаткові заходи щодо виправлення помилок та тестування. Вони також повинні бути описані в даному пункті пояснювальної записки.

Далі наведені методичні рекомендації щодо тестування.

Тестування програми – процес виконання програмного коду, спрямований на виявлення існуючих у ньому дефектів. Під дефектом розуміється ділянка програмного коду, виконання якої за певних умов приводить до несподіваного поводження системи (тобто поводження, що не відповідає вимогам).

Завдання тестування – визначення умов, при яких проявляються дефекти системи, і протоколювання цих умов.

Мета застосування процедури тестування програмного коду – мінімізація кількості дефектів у кінцевому продукті.

Тестування саме по собі не може гарантувати повної відсутності дефектів у програмному коді системи. Однак, у сполученні із процесами верифікації й валідації, спрямованими на усунення суперечливості й неповноти проектної документації (зокрема – вимог на систему), грамотно

організоване тестування дає гарантію того, що система задовольняє вимогам і поводиться відповідно до них у всіх передбачених ситуаціях.

Основні методи тестування

"Чорний ящик"

Основна ідея тестування системи як "чорного ящика" полягає в тому, що тестувальнику доступні тільки вимоги на систему, що описують її поводження, і сама система. Всі внутрішні особливості реалізації системи сховані. Аналіз системи полягає в подачі на її "вхід" деяких зовнішніх впливів і спостереженні результату, що з'являється на її "виході".

У програмній системі "чорний ящик" може становити набір класів (або модулів) з відомими зовнішніми інтерфейсами, але недоступними вихідними текстами.

Тестування за методом "чорного ящика" називають також тестуванням за вимогами.

"Скляний (білий) ящик"

При тестуванні програмної системи як "скляного ящика" тестувальник має доступ не тільки до вимог до системи, її входів і виходів, але й до її внутрішньої структури – програмному коду. Доступність програмного коду розширює можливості тестування.

Оскільки сучасні програмні системи мають досить значні розміри, при тестуванні їхнього програмного коду використовується метод функціональної декомпозиції. Система розбивається на окремі модулі (класи, простори імен і т. ін.), що мають певну функціональність і інтерфейси. Після цього окремо тестується кожний модуль – виконується модульне тестування. Потім відбувається об’єднання окремих модулів у більші конфігурації – виконується інтеграційне тестування, і нарешті, тестується система в цілому – виконується системне тестування.

Модульне тестування

У ході модульного тестування кожний модуль тестується як на відповідність вимогам, так і на відсутність проблемних ділянок програмного коду, які можуть викликати відмови й збої в роботі системи. Як правило, модулі не працюють поза системою – вони приймають дані від інших модулів, переробляють їх і передають далі. Для того, щоб з одного боку, ізолювати модуль від системи й виключити вплив потенційних помилок системи, а з іншого боку – забезпечити модуль всіма не обхідними даними, використовується тестове оточення.

Завдання тестового оточення – створити середовище виконання для модуля, емулювати всі зовнішні інтерфейси, до яких звертається модуль.

Типова процедура тестування складається в підготовці й виконанні тестових прикладів. Кожний тестовий приклад перевіряє одну "ситуацію" у поводженні модуля й складається зі списку значень, переданих на вхід модуля, опису запуску й виконання переробки даних – тестового сценарію й списку значень, які очікуються на виході модуля у випадку його коректного поводження. Тестові сценарії складаються таким чином, щоб виключити звертання до внутрішніх даних модуля, вся взаємодія повинна відбуватися тільки через його зовнішні інтерфейси.

Виконання тестового приклада підтримується тестовим оточенням, що містить у собі програмну реалізацію тестового сценарію. Виконання починається з передачі модуля вхідних даних і запуску сценарію. Реальні вихідні дані, отримані від модуля в результаті виконання сценарію, зберігаються й порівняються з очікуваними. У випадку їхнього збігу тест вважається пройденим, у іншому випадку – не пройденим.

Результатом тестування й верифікації окремих модулів, що становлять програмну систему, повинен бути висновок про те, що ці модулі є внутрішньо несуперечливими й відповідають вимогам.

Інтеграційне тестування

Окремі модулі рідко функціонують самі по собі, тому наступне завдання після тестування окремих модулів – тестування коректності взаємодії декількох модулів, об'єднаних у єдине ціле. Таке тестування називають інтеграційним.

Інтеграційне тестування називають ще тестуванням архітектури системи.

З одного боку, ця назва обумовлена тим, що інтеграційні тести містять у собі перевірки всіх можливих видів взаємодій між програмними модулями й елементами, які визначені в архітектурі системи. Таким чином, інтеграційні тести перевіряють повноту взаємодій у реалізації системи, що тестується.

З іншого боку, результати виконання інтеграційних тестів є одним з основних джерел інформації для процесу поліпшення й уточнення архітектури системи, міжмодульних та міжкомпонентних інтерфейсів. Із цього погляду, інтеграційні тести перевіряють коректність взаємодії компонент системи.

Системне тестування

По завершенню інтеграційного тестування всі модулі системи є погодженими по інтерфейсах і функціональності. Починаючи із цього моменту можна переходити до системного тестування, тобто тестування поводження системи в цілому як єдиного об'єкта.

При системному тестуванні проводиться не тільки функціональне тестування, але й оцінка характеристик якості системи – її стійкості, надійності, безпеки й продуктивності. На цьому етапі виявляються багато проблем зовнішніх інтерфейсів системи, пов'язаних з неправильною взаємодією з іншими системами, апаратним забезпеченням, неправильним розподілом пам'яті, відсутністю коректного звільнення ресурсів і т. ін.

Як правило, для системного тестування застосовується метод "чорного ящика", при цьому як вхідні та вихідні дані використовуються реальні дані, з якими працює система, або дані, подібні їм.

Вхідною інформацією для проведення системного тестування є два класи вимог: функціональні й нефункціональні.

Функціональні вимоги явно описують, що система повинна робити і які перетворення вхідних даних виконувати.

Нефункціональні вимоги визначають властивості системи, прямо не пов'язані з її функціональністю. Прикладом таких властивостей може служити максимальний час відгуку системи на запит користувача, мінімальний час безперебійної роботи системи й ін.

Системне тестування проводиться в кілька етапів, на кожному з яких використовується один з видів системного тестування.

Види системного тестування:

1. Функціональне тестування

Даний вид тестування призначений для підтвердження того, що вся система в цілому поводиться відповідно до очікувань користувача, формалізованих у вигляді системних вимог. У ході функціонального тестування перевіряються всі функції системи з погляду її користувачів (як людей, так і інших програмних систем). Також необхідно перевірити функціональну повноту користувальницького інтерфейсу й коректність виведення інформації.

При функціональному тестуванні система розглядається як "чорний ящик".

Критерії повноти функціонального тестування:

1. Всі функціональні вимоги повинні бути протестовані.

2. Всі класи припустимих вхідних даних повинні коректно оброблятися системою.

3. Всі класи неприпустимих вхідних даних повинні бути відкинуті системою, при цьому не повинна порушуватися стабільність її роботи.

4. У тестових прикладах повинні генеруватися всі можливі класи вихідних даних системи.

5. Під час тестування система повинна побувати у всіх своїх внутрішніх станах, пройшовши при цьому по всіх можливих переходах між станами.

2. Тестування продуктивності

Даний вид тестування спрямований на визначення того, що система забезпечує належний рівень продуктивності при обробці користувальницьких запитів. Виділяють три основних фактори, що впливають на продуктивність системи: кількість підтримуваних системою потоків (наприклад, користувальницьких сесій), кількість вільних системних ресурсів, кількість вільних апаратних ресурсів.

Тестування продуктивності дозволяє виявляти вузькі місця в системі, які проявляються в умовах підвищеного навантаження або недостачі системних ресурсів. У цьому випадку за результатами тестування проводиться доробка системи, змінюються алгоритми виділення й розподілу ресурсів системи.

Всі вимоги, що ставляться до продуктивності системи, повинні бути чітко визначені й обов'язково повинні містити в собі числові оцінки параметрів продуктивності. Наприклад, вимога "Система повинна мати прийнятний час відгуку на запит користувача" є непридатною для тестування. Навпаки, вимога "Час відгуку на запит користувача не повинен перевищувати 2 секунди" може бути протестована.

У звіті за даним видом тестування вказують такі показники, як завантаження апаратного й системного програмного забезпечення (кількість циклів процесора, виділеної пам'яті, кількість вільних системних ресурсів і т. ін.). Також важливі швидкісні характеристики системи, що тестується: кількість оброблених за одиницю часу запитів, часові інтервали між початком обробки кожного наступного запиту, рівномірність часу відгуку в різні моменти часу й т. ін.

3. Навантажувальне тестування

Має багато загального з тестуванням продуктивності, однак його основне завдання – оцінити продуктивність і стійкість системи у випадку, коли для своєї роботи вона виділяє максимально доступну кількість ресурсів або коли вона працює в умовах їхньої критичної недостачі.

Основна мета навантажувального тестування – вивести систему з ладу, визначити ті умови, при яких вона не зможе далі нормально функціонувати.

Навантажувальне тестування дуже важливе при тестуванні Web-систем, рівень навантаження на які найчастіше дуже складно прогнозувати.

4. Тестування конфігурації

Незважаючи на те, що в теперішній час особливості реалізації периферійних пристроїв приховуються відповідними драйверами операційних систем, проблеми сумісності (як програмної, так і апаратної) однаково існують.

У ході тестування конфігурації перевіряється, що програмна система коректно працює на всьому підтримуваному апаратному забезпеченні й разом з іншими програмними системами.

Також необхідно перевіряти, що система коректно обробляє проблеми, що виникають в устаткуванні, як штатні (наприклад, сигнал кінця паперу в принтері), так і позаштатні (збій живлення).

5. Тестування безпеки

Якщо програмна система призначена для зберігання або обробки даних, які становлять таємницю певного роду (наприклад, комерційну), то до властивостей такої системи, пред'являються підвищені вимоги. Виконання цих вимог перевіряється при тестуванні безпеки системи.

При використанні цього виду тестування перевіряється:

1. Організація авторизації й аутентифікації користувачів.
2. Коректність реєстрації в "журналі аудита" всіх подій системи, пов'язаних з безпекою.
3. Архітектура системи з погляду забезпечення захисту інформації від несанкціонованого доступу.
4. Наявність і повноту опису засобів забезпечення безпеки в документації на програмну систему.

6. Тестування надійності й відновлення після збоїв

Для коректної роботи системи в будь-якій ситуації необхідно впевнитися в тому, що вона відновлює свою функціональність і продовжує коректно працювати після будь-якої проблеми, що перервала її роботу.

При тестуванні відновлення після збоїв імітуються збої устаткування, що оточує програмне забезпечення або збої програмної системи, викликані зовнішніми факторами. При аналізі поводження системи в цьому випадку необхідно звертати увагу на два фактори – мінімізацію

втрат даних у результаті збоїв й мінімізацію часу між збоєм і продовженням нормального функціонування системи.

7. Тестування зручності використання

Зручність використання визначає ступінь простоти доступу користувача до функцій системи, надаваним через користувальницький інтерфейс.

Зручність використання користувальницького інтерфейсу – показник його якості, що визначає кількість зусиль, необхідних для вивчення принципів роботи із програмною системою за допомогою даного інтерфейсу, її використання, підготовки вхідних даних і інтерпретації вихідних даних.

Як правило, при тестуванні зручності використання користувальницького інтерфейсу використовуються деякі евристичні критерії й характеристики (див. пункт 2.2.1).

У результаті виконання всіх розглянутих раніше видів тестування робиться висновок про функціональність і властивості системи, після чого вузькі місця системи допрацьовуються до реалізації необхідної функціональності або до досягнення системою необхідних властивостей.

У ході системного тестування не завжди застосовуються всі з перерахованих видів тестування. Конкретний їхній набір залежить від конкретної програмної системи.

Документування процедури тестування

Основне призначення документації, створюваної при тестуванні, – забезпечення гарантій того, що процес тестування виконується з необхідною якістю й всі аспекти поводження системи протестовані.

Перелік необхідної документації:

1. Тест-вимоги.
2. Тест-план.
3. Звіт про тестування.

Тест-вимоги розробляються на підставі системних і функціональних вимог до додатка. У них докладно описується, які аспекти поводження системи повинні бути протестовані, щоб упевнитися в її коректному функціонуванні, і на підставі якого зовнішнього ефекту можна переконатися, що функціональність, яка перевіряється, реалізована правильно.

Тест-вимоги повинні бути достатніми для побудови тест-плану перевірки програмної системи без знайомства з її програмним кодом.

Структура тест-вимог повинна додержуватися структури розділу функціональних вимог на систему. Як правило, одній системній або функціональній вимозі відповідає мінімум одна тест-вимога.

Класифікація тест-вимог за призначенням:

1. Контроль вхідних даних.
2. Обробки помилок уведення даних і обробки інформації.
3. Одержання основного результату.
4. Оформлення й виведення результатів.

Для кожної тест-вимоги повинна існувати можливість перевірки – виконується ця вимога в реалізованій системі чи ні.

На підставі тест-вимог створюються тест-план – документ, що містить докладний покроковий опис того, як повинні бути протестовані тест-вимоги. На відміну від тест-вимог, у тест-плані описуються конкретні способи перевірки функціональності системи.

Як правило, тест-план складається з окремих тестових прикладів, кожний з яких перевіряє деяку функцію або набір функцій системи. Для кожного тестового приклада однозначно визначається критерій успішного проходження, за допомогою якого можна судити про відповідність поводження системи заданому.

Структура тест-плану повинна відповідати структурі тест-вимог.

Кожний пункт тест-плану повинен містити:

1. Посилання на вимогу(и), що перевіряється цим пунктом;
2. Конкретне значення вхідних даних.
3. Очікувану реакцію програми (тексти повідомлень, значення результатів).
4. Опис послідовності дій, необхідних для виконання пунктів тест-плану.

При ручному тестуванні зручним є подання тест-плану у вигляді текстового документа, у якому окремі розділи становлять описи тестових прикладів. Кожний тестовий приклад повинен містити в собі перерахування послідовності дій, які необхідно виконати для проведення тестування, а також очікувані відгуки системи на ці дії.

За результатами виконання тестів створюється звіт про виконання тестування. Він є основним джерелом для висновку про ступінь відповідності протестованої системи вимогам. Такий звіт як мінімум повинен містити інформацію про кожний виконаний тестовий приклад і результат його виконання (успіх або невдача).

Іноді тест-план сполучають зі звітом про проведення тестування, додаючи до нього інформацію про отриману реакцію системи й збіг (розбіжності) отриманих результатів з очікуваними. Наприкінці опису кожного тестового приклада додається інформація про те, чи пройдений тестовий приклад у цілому. Наприкінці всього тест-плану, сполученого зі звітом, міститься графа "Тестових прикладів пройдене/усього", у яку заноситься число пройдених тестових прикладів і загальна їхня кількість.

Приклад тест-плану, сполученого зі звітом про проведення тестування наведений у додатку Ж.

Пункт 2.2.4. Контрольний приклад

Працездатність додатка перевіряється за результатами його випробувань на контрольному прикладі.

Контрольний приклад складається з урахуванням необхідності перевірки всіх вимог до додатка, описаних у відповідній специфікації. Він повинен містити інформацію, достатню для перевірки всіх основних функцій розробленої програми, які забезпечують реалізацію бізнес-завдань модуля. У контрольному прикладі повинні використовуватися реальні масиви даних, пов'язані з конкретними умовами експлуатації додатка.

У даному пункті пояснювальної записки необхідно привести опис контрольного прикладу, правил його запуску й виконання:

1. Призначення контрольного прикладу.
2. Функції програми, що підлягають перевірці.

Повинен містити коротку характеристику функцій із числа реалізованих програмою, які перевіряються контрольним прикладом.

3. Вхідні дані для перевірки (таблиці бази даних з оперативними даними, довідники – таблиці нормативно-довідкової інформації).
4. Порядок перевірки програми.

Повинен містити:

опис складу технічних засобів, необхідних для роботи програми або посилання на відповідні програмні документи;

опис процедур запуску програми, яка перевіряється;

опис результатів роботи програми, яка перевіряється, отриманих на вхідних даних контрольного прикладу.

Розділ **"Висновки"** повинен містити висновки щодо розроблених проектних рішень, параметрів проектованого модуля, розробки рекомен-

дацій щодо конкретного використання проектних рішень, які переваги отримає бізнес від використання програмного продукту.

Розділ **"Використана література"** повинен містити відомості про літературні джерела, використані при розробці проекту.

Розділ **"Додатки"** повинен містити контрольний приклад з формами заповнених вхідних і вихідних документів, машинограм, роздруківки відеокадрів, лістинги програм. Обов'язково подати отримані на ПК вихідні машинограми.

5. Методичні рекомендації щодо оформлення проекту

Важливе значення при роботі над комплексним курсовим проектом має його оформлення, до якого викладаються певні вимоги. Весь матеріал курсового проекту треба розташувати в певній послідовності. Титульний аркуш оформляється за встановленою формою.

У змісті подаються заголовки розділів, підрозділів, пунктів із зазначенням сторінок, з яких вони починаються. При цьому заголовки повинні бути наведені в чіткій відповідності з текстом.

Текстовий матеріал комплексного курсового проекту друкується на ПК на папері стандартного формату (210 х 297). Текст повинен відповідати правилам граматики й стилістики.

При написані текстового матеріалу сторінки повинні бути відформатовані таким чином: ліве поле – не менше 30 мм, праве – не менше 10 мм, верхнє – не менше 15 мм, нижнє – не менше 20 мм.

Абзац повинен починатися з відстані 35 мм від лівого краю сторінки.

Не дозволяється розміщати заголовки й підзаголовки в нижній частині сторінки, якщо на ній не більше 4 рядків наступного тексту.

Кожний розділ курсового проекту повинен починатися з нової сторінки, назви підрозділів, пунктів, підпунктів, параграфів – з абзацу.

Підкреслення найменувань розділів, підрозділів, пунктів не допускається. Відстань між заголовками розділів, підрозділів, пунктів і наступним текстом повинна бути на 5 мм більше відстані між рядками тексту. Заголовки структурних елементів проекту "ЗМІСТ", "ВСТУП", "ВИСНОВКИ", "ВИКОРИСТАНА ЛІТЕРАТУРА" і заголовки розділів треба писати великими друкованими літерами без крапки наприкінці. При друкуванні назви розділів центруються.

Заголовки підрозділів, пунктів і підпунктів треба починати з великої літери також без крапки наприкінці. Переноси в середині слова в заголовках не допускаються.

Розділи, підрозділи, пункти, підпункти проекту треба нумерувати арабськими цифрами. Розділи мають порядкову нумерацію, наприклад: 1., 2., 3. і т. д. Підрозділи повинні мати порядкову нумерацію в межах розділу. Номер підрозділу включає номер розділу й порядковий номер підрозділу, які розділяються крапкою, наприклад: 1.1., 12., 1.3. і т. д. Номер пункту включає номер розділу, підрозділу, порядковий номер пункту і розділяються вони крапкою, наприклад: 1.1.1., 1.1.2., 1.1.3. і т. д.

Сторінки курсового проекту повинні бути пронумеровані арабськими цифрами в правому верхньому куті без крапки. Нумерація сторінок наскрізна від титульного аркуша до останнього аркуша тексту, включаючи ілюстрації, таблиці, графіки. На титульному аркуші, у завданні на курсовий проект і змісті нумерація сторінок не проставляється.

Викладені у тексті матеріали повинні наочно доповнювати й підтверджувати ілюстрації (схеми, рисунки, графіки, діаграми). Ілюстрації повинні відбивати тему курсового проекту. Студентові необхідно продумати, який матеріал проілюструвати. Це діаграми аналізу бізнес-процесів, схеми алгоритмів, схеми інформаційних зв'язків тощо.

Усі ілюстрації іменуються рисунками, позначаються словом "Рис", їм привласнюється порядковий номер (у межах номера розділу). Рисунки треба виконувати на одній сторінці й розташовувати відразу після згадування в тексті. Якщо рисунок не вміщається на одній сторінці, його можна переносити на інші сторінки, при цьому на кожній наступній сторінці вказується "Рис, продовження".

На всі таблиці повинні бути посилання. Розташовувати їх треба безпосередньо після тексту, де вони згадуються вперше або на наступній сторінці. Нумеруються таблиці послідовно в межах розділу проекту. Над правим верхнім кутом таблиці міститься напис "Таблиця" із зазначенням її порядкового номера. Таблиця повинна мати найменування, яке розташовується на наступному рядку після слова "Таблиця".

Перерахування, при необхідності, можуть бути наведені усередині пунктів, їх варто нумерувати порядковою нумерацією арабськими цифрами з дужкою й писати малими літерами з абзацу.

Формули треба виділяти з тексту в окремий рядок, залишаючи нижче й вище формули по одному вільному рядку. Формули треба нумерувати

порядковою нумерацією в межах розділу проекту арабськими цифрами в круглих дужках у крайньому правом положенні на рядку. Пояснення значень символів числових коефіцієнтів формули треба подавати безпосередньо під формулою у тій же послідовності, в якій вони в ній дані.

Значення кожного символу й числового коефіцієнта необхідно давати з нового рядка. Перший рядок пояснення починають зі слова "де" без двокрапки.

Використані в процесі роботи над комплексним курсовим проектом спеціальні літературні джерела вказуються наприкінці проекту перед додатком. Літературні джерела треба проводити за абеткою.

Кожне літературне джерело відбивається в списку в такому порядку: порядковий номер у списку, прізвище й ініціали автора, назва книги (для статті – її заголовок, назва збірника, журналу, його номер), видавництво, місце й рік випуску. Наприклад: Вигерс Карл "Разработка требований к программному обеспечению : пер. с англ. / Карл Вигерс. – М. : Изд. дом "Русская редакция", 2004. – 576 с.

При посиланні на літературне джерело в тексті наводиться його порядковий номер, записаний у квадратні дужки, а через кому – номер сторінки. Наприклад: [14, с. 38].

Додаток треба оформляти як продовження проекту. Кожний додаток повинен починатися з нової сторінки й мати змістовний заголовок, написаний великими друкованими літерами. У правому верхньому куті над заголовком повинно бути написане слово "Додаток". Додатки позначаються послідовно літерами: А, Б, В, Г, Д, Ж, З, К, Л, М, за винятком букв Ґ Є, І, Ї, Й, О, Ч, Ь українського алфавіту.

Наявні в додатку рисунки й таблиці, треба нумерувати в межах кожного додатка, наприклад: Рис. А1. Завдання на курсове проектування виконується відповідно до **додатка Б** титульний аркуш оформлюється відповідно до **додатка Е**.

Рекомендована література

1. Бондаренко М. Ф. Моделирование и проектирование бизнес-систем: методы, стандарты, технологии : учеб. пособ. / М. Ф. Бондаренко, С. И. Маторин, Е. А. Соловьев. – Харьков : Компания СМИТ, 2004. – 272 с.

2. Бутова Р. К. Інформаційні системи в промисловості : конспект лекцій / Р. К. Бутова. – Харків : Вид. ХДЕУ, 2004. – 176 с.

3. Бутова Р. К. Система оброблення економічної інформації : конспект лекцій / Р. К. Бутова. – Харків : Вид. ХНЕУ, 2005. – 220 с.

4. Гужва В. М. Інформаційні системи і технології на підприємствах : навч. посіб. / В. М. Гужва. – К. : КНЕУ, 2001. – 400 с.

5. Єрьоміна Н. В. Банківські інформаційні системи / Н. В. Єрьоміна. – К. : КНЕУ, 2000. – 220 с.

6. Золотарьова І. О. Інформаційні системи та технології в банківській сфері : навчальний посібник / І. О. Золотарьова, Р. К. Бутова, А. А. Гаврилова. – Харків : Вид. ХНЕУ, 2009. – 332 с. (укр. мов.).

7. Інформатизація бізнесу: концепції, технології, системи / А. М. Карминский, С. А. Карминский, В. П. Нестеров, Б. В. Чернишов; під ред. А. М. Карминского. – М. : Фінанси й статистика, 2004. – 624 с.

8. Інформаційні системи і технології в економіці / за ред. д. е. н. В. С. Пономаренка. – К. : Видавничий центр "Академія", 2002. – 542 с.

9. Інформаційні системи у фінансово-кредитних установах : навч.-метод. посібн. для самост. вивч. дисц. / І. Ф. Рогач, М. А. Сензюк, В. А. Антонюк, О. О. Денісова. – К. : КНЕУ, 2001. – 324 с.

10. Информационные технологии в бизнесе : энциклопедия; пер. с англ. / под ред. М. Желены. – СПб. : Питер, 2002. – 1120 с.

11. Информационные технологии в кадровом учете и управлении персоналом 2007. Аналитический обзор российского рынка автоматизированных систем управления персоналом. – М. : Tadviser, 2007. – 81 с.

12. Информационные технологии в маркетинге : учебник для вузов / Г. А. Титоренко, Г. Л. Макарова, Д. М. Дайитбегов и др. ; под ред. проф. Г. А. Титоренко – М. : ЮНИТИ-ДАНА, 2000. – 335 с.

13. Информационные технологии в экономике / под ред. д. э. н., профессора Ю. Ф. Симионова. Серия Высшее образование. – Ростов н/Д : Феникс, 2003. – 352 с.

14. Леоненков А. В. Самоучитель UML. [Текст] / А. В. Леоненков. – СПб. : Бхв-Петербург, 2004. – 432 с.

15. Лодон Дж. Управление информационными системами / Дж. Лодон, К. Лодон; пер. с англ. под. ред. Д. Р. Трутнева. – 7-е изд. – СПб. : Питер, 2005. – 912 с.

16. Методичні рекомендації до оформлення звітів, курсових та дипломних проектів для студентів напряму підготовки 0804 "Комп'ютерні науки" всіх форм навчання / укл.: І. О. Золотарьова, О. М. Беседовський, І. Л. Латишева, Г. О. Плеханова. – Харків : Вид. ХНЕУ, 2007. – 32 с. (укр. мов.)

17. Пинчук Н. С. Інформаційні системи й технології в маркетингу : навч. посібн. – 2-ге вид., перероб. і доп. / Н. С. Пинчук, Г. П. Галузинський, Н. С. Орленко – К. : КНЕУ, 2003. – 352 с.

18. Писаревська Т. А. Інформаційні системи і технології в управлінні трудовими ресурсами : навч. посіб. / Т. А. Писаревська. – 2-ге вид., перероб. і доп. – К. : КНЕУ, 2000. – 279 с.

19. Татарчук М. І. Корпоративні інформаційні системи : навч. посібн. / М. І. Татарчук. – К. : КНЕУ, 2005. – 291 с.

20. Терещенко Л. О. Інформаційні системи і технології в обліку : навч. посібн. / Л. О. Терещенко, І. І. Матієнко-Зубенко. – К. : КНЕУ, 2004. – 187 с.

21. Успенский И. В. Интернет как инструмент маркетинга. – СПб. : БХВ-Санкт-Петербург, 2000. – 256 с.

22. Уткин В. Б. Информационные системы в экономике : учебник для студ. высш. учебн. заведений / В. Б. Уткин, К. В. Балдин. – М. : Издательский центр "Академия", 2004. – 288 с.

23. Чаадаев В. К. Информационные системы компаний связи. Создание и внедрение / под. ред. : И. В. Шеметова, И. Б. Шибаева. – М. : Эко-Трендз, 2004. – 256 с.

Ресурси мережі Інтернет

24. Автоматизированная Банковская система "БИС ГРАНТ". – Режим доступа : http://www. bank soft. com.ua/index.php?id=9

25. Всё о разработке ПО. – Режим доступа : http://www.maxkir.com/

26. Всё о CRM в России и СНГ. – Режим доступа : http://www. CRMinfo.ru

27. Выбор КИС: проблемы и решения. – Режим доступа : http://soft-expert.ru/

28. Издание о высоких технологиях – CNews. – Режим доступа : http://www.cnews.ru/

29. Институт искусственного интеллекта. – Режим доступа : http://www.iai.gov.ua/

30. Интернет-университет информационных технологий. – Режим доступа : http://www.intuit.ru/

31. Информатика – ГОСНИИ ИТ. – Режим доступа : http://www. informika.ru/

32. Информационные технологии в управлении. – Режим доступа : http://www.it-management.ru

33. Информационный портал CRM. – Режим доступа : www.crm.com.ua

34. IT-портал. – Режим доступа : http://www.citforum.ru/

35. Компьютерная библиотека. – Режим доступа : http://www. computerlibrary.info

36. Корпоративное управление. – Режим доступа : http://www. corporation.com.ua/

37. Корпоративный менеджмент. – Режим доступа : http://www.cfin.ru/

38. Корпорация "Галактика". Информационные технологи управления. – Режим доступа : http://www.galaktika.ru/

39. Леоненков А. В. Визуальное моделирование в среде IBM Rational Rose 2003 [Электронный ресурс] / А. В. Леоненков. Режим доступа : http://www.intuit.ru/department/se/ibmrrose/

40. Леоненков А. В. Нотация и семантика языка UML [Электронный ресурс] / А. В. Леоненков. – Режим доступа : http://www.intuit.ru/department/pl/umlbasics/

41. Открытые системы. – Режим доступа : http://www.osp.ru/

42. Планета КИС. – Режим доступа : http://www.russianenterprise-solutions.com

43. Портал "Информационно-коммуникационные технологии в образовании". – Режим доступа : http://www.ict.edu.ru/

44. Портал "Компьютерра онлайн". – Режим доступа : http://www.computerra.ru

45. ПРОФЕССИОНАЛ УПРАВЛЕНИЯ ПРОЕКТАМИ. – Режим доступа : http://www.pmprofy.ru/

46. Российская Ассоциация Управления Проектами СОВНЕТ. – Режим доступа : http://www.sovnet.ru/

47. Сокур С. Как строить систему управления: прагматичный под.-ход [Электронный ресурс] / С. Сокур, О. Коваленко. – Режим доступа : http://www.management.com.ua.

48. Управление проектами. – Режим доступа : http://www.pmprofy.ru/

49. Управление проектами в России. – Режим доступа : http://www.projectmanagement.ru/

50. BYTE-Россия – журнал для ИТ-профессионалов. – Режим доступа : http://www.bytemag.ru/

51. ERP-эксперт – Всё о ERP, ERP II, MRP, MRP II. – Режим доступа : http://erp-expert.narod.ru/index.htm

52. Effective Business Modeling with UML: Describing Business Use Cases and Realizations. – Access mode : http://www.ibm.com/developerworks/rational/library/content/RationalEdge/nov02/BusinessModelingWithUML_TheRationalEdge_Nov2002.pdf

53. Introduction to the Unified Modeling Language [Electronic resource]. – Access mode: **Ошибка! Недопустимый объект гиперссылки.**

54. ITC Online. – Режим доступа : http://itc.ua/

55. The official UML Web site. – Access mode: http://www.uml.org

56. Ten Usability Heuristics // Web-сайт Якоба Нильсена. – Режим доступу : http://www.useit.com/papers/ heuristic/heuristic_list.html/ – 10.04.2010 р. – Загол. з екрану.

57. UML basics: An introduction to the Unified Modeling Language. [Electronic resource] – Access mode : http://www.ibm.com/developerworks/rational/library/content/03July/2500/2772

58. http://idefdfd.ru/

59. http://www.idef.com/Downloads.htm

60. http://194.44.242.244/e-journals/DeBu/2008-1/doc/1/17.pdf

Додатки

Додаток А

Тематика комплексного курсового проекту

1. Розроблення модулів ІС підприємства на основі WEB-технологій та мобільних технологій.

1.1. Розроблення модуля "Управління заказами клієнтів" на основі WEB-технологій.

1.2. Розроблення модуля "Управління продажами продукції" на основі WEB-технологій.

1.3. Розроблення модуля "Управління продажами продукції" на основі мобільних технологій.

1.4. Розроблення модуля "Управління взаємодіями з клієнтами" на основі WEB-технологій.

1.5. Розроблення модуля "Управління взаємодіями з партнерами по бізнесу" на основі WEB-технологій.

1.6. Розроблення модуля "Управління взаємодіями з постачальниками" на основі WEB-технологій.

1.7. Розроблення модуля "Управління співпрацею працівників підприємства" у середовищі корпоративного інформаційного порталу.

1.8. Розроблення модуля "Управління співпрацею з контрагентами" у середовищі корпоративного інформаційного порталу.

1.9. Розроблення модуля "Організація бази знань" у середовищі корпоративного порталу знань.

1.10. Розроблення модуля "Аналіз місткості ринку продукції" на основі WEB-технологій.

1.11. Розроблення модуля "Стратегічний розвиток бізнесу підприємства" на основі WEB-технологій.

1.12. Розроблення модуля "Просування продукції підприємства" на основі WEB-технологій.

1.13. Розроблення модуля "Управління контактами з клієнтами підприємства" на основі WEB-технологій.

1.14. Розроблення модуля "Маркетингові дослідження сегменту ринку" на основі WEB-технологій.

1.15. Розроблення модуля "Підбір кадрів для підприємства" на основі WEB-технологій.

Продовження додатка А

1.16. Розроблення модуля "Мобільний офіс для співробітників підприємства" на основі мобільних та інтернет-технологій.

2. Розроблення модулів ІС маркетингу та систем управління взаєминами з клієнтами.

2.1. Розроблення автоматизованого модуля "Управління рекламними заходами на підприємстві".

2.2. Розроблення автоматизованого модуля "Оцінка ефективності рекламних заходів на підприємстві".

2.3. Розроблення автоматизованого модуля "Аналіз попиту на товари підприємства".

2.4. Розроблення автоматизованого модуля "Оцінка кон'юнктури товарного ринку".

2.5. Розроблення автоматизованого модуля "Маркетингові дослідження товару".

2.6. Розроблення автоматизованого модуля "Аналіз рівня конкурентоспроможності товару".

2.7. Розроблення операційного модуля CRM-системи підприємства.

2.8. Розроблення модуля CRM-системи "Аналіз клієнтської бази підприємства".

2.9. Розроблення коллаборaційного модуля CRM-системи підприємства.

2.10. Розроблення мобільного модуля CRM-системи підприємства.

2.11. Розроблення модуля CRM-системи "Управління сервісним обслуговуванням продукції підприємства".

2.12. Розроблення модуля CRM-системи "Аналіз лояльності клієнтів до підприємства та формування бази потенційних клієнтів".

3. Розроблення модулів ІС управління персоналом.

3.1. Розроблення автоматизованого модуля "Ділова оцінка персоналу підприємства".

3.2. Розроблення автоматизованого модуля "Планування кар'єри персоналу підприємства".

3.3. Розроблення автоматизованого модуля "Формування кадрового резерву підприємства".

Продовження додатка А

3.4. Розроблення автоматизованого модуля "Аналіз укомплектованості штату підприємства".

3.5. Розроблення автоматизованого модуля "Управління навчанням та підвищення кваліфікації персоналу".

3.6. Розроблення автоматизованого модуля "Атестація персоналу".

4. Розроблення модулів автоматизованої банківської системи.

4.1. Розроблення автоматизованого модуля "Управління вкладними операціями".

4.2. Розроблення автоматизованого модуля "Управління кредитними операціями фізичних осіб".

4.3. Розроблення автоматизованого модуля "Управління кредитними операціями юридичних осіб".

4.4. Розроблення автоматизованого модуля "Аналіз кредитоспроможності позичальників – юридичних осіб".

4.5. Розроблення автоматизованого модуля "Аналіз кредитоспроможності позичальників – фізичних осіб".

4.6. Розроблення автоматизованого модуля "Моніторинг кредитного портфеля банку".

4.7. Розроблення автоматизованого модуля "Ведення корпоративних карткових рахунків клієнтів банку".

4.8. Розроблення автоматизованого модуля "Зарплатний проект" на базі карткових технологій.

4.9. Розроблення автоматизованого модуля "Ведення карткових рахунків клієнтів – фізичних осіб".

4.10. Розроблення автоматизованого модуля "Приймання та облік комунальних платежів" на основі WEB-технологій.

4.11. Розроблення автоматизованого модуля "Управління операціями обмінного пункту валют".

5. Розроблення модулів систем електронного документообігу.

5.1. Розроблення модуля "Електронна канцелярія".

5.2. Розроблення модуля "Електронний офіс фірми".

5.3. Розроблення модуля "Формування доручень та контроль їх виконання в СЕД".

5.4. Розроблення модуля "Реєстрація та маршрутизація вхідних документів в СЕД".

Закінчення додатка А

5.5. Розроблення модуля "Пошук документів в електронному архіві".

5.6. Розроблення модуля "Управління WEB-контентом сайта".

5.7. Розроблення модуля "Управління спільною роботою віртуально розподіленої команди".

5.8. Розроблення модуля "Інтеграція додатків через єдиний інтерфейс" на базі корпоративного інформаційного порталу.

5.9. Розроблення модуля "Забезпечення криптографічного захисту електронного документообігу".

5.10. Розроблення модуля "Аутентифікація паперового документу з забезпеченням його дійсності".

6. Розроблення модулів ІС з управління діяльністю підприємства.

6.1. Розроблення автоматизованого модуля "Управління договорами на поставку продукції".

6.2. Розроблення автоматизованого модуля "Формування планів поставок продукції покупцям".

6.3. Розроблення автоматизованого модуля "Формування портфеля замовлень виробництву на планований період".

6.4. Розроблення автоматизованого модуля "Формування виробничої програми випуску продукції на планований період".

6.5. Розроблення автоматизованого модуля "Формування детального плану виробництва виробничим цехам на планований період".

6.6. Розроблення автоматизованого модуля "Облік випуску готової продукції за звітний період".

6.7. Розроблення автоматизованого модуля "Аналіз виконання договірних зобов'язань за звітний період".

6.8. Розроблення автоматизованого модуля "Календарне планування відвантаження продукції покупцям на планований період".

6.9. Розроблення автоматизованого модуля "Складський облік товарів за звітний період".

6.10. Розроблення автоматизованого модуля "Облік витрат на виробництво за звітний період".

6.11. Розроблення автоматизованого модуля "Управління замовленнями споживачів продукції".

6.12. Розроблення автоматизованого модуля "Аналіз фінансового стану підприємства" за звітний період.

Додаток Б

Зразок оформлення завдання на комплексне курсове проектування

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Кафедра інформаційних систем
Спеціальність 7.080401
курс 5, група 6, семестр 9

ЗАВДАННЯ
на комплексний курсовий проект
студента
Іванова Віктора Михайловича

1. Тема комплексного курсового проекту: "Розроблення автоматизованого модуля "Оцінка ефективності рекламних заходів на підприємстві".

2. Строк здачі студентом закінченого проекту: 15.12.2010 р.

3. Вхідні дані до проекту: ДСТУ з обробки інформації, матеріали проектно-технологічної практики, готові програмні продукти, технічна документація на АІС <назва об’єкта управління>, літературні джерела, інтернет-видання.

4. Зміст пояснювальної записки: Вступ. Розділ 1. Постановка завдань дослідження та проектування модуля <назва модуля>. Розділ 2. Проектні й технічні рішення та документація. Висновки.

5. Перелік графічного матеріалу: діаграми IDEF0, DFD; схема інформаційних зв'язків; UML-діаграми.

Дата видачі завдання: 10.09.2010 р.

Керівник проекту к.е.н., доц.

Студент Іванов В. М.

Додаток В

Приклад оформлення анотації до комплексного курсового проекту

Анотація
до теми комплексного курсового проекту
"Розробка автоматизованого модуля "Планування кар'єри"
з дисципліни "ІС в сучасному бізнесі"

Ключові моменти проекту :

1. Існуюча **бізнес-проблема**, на рішення якої спрямована ІС, що розробляється: проаналізувати кадровий склад персоналу та укомплектувати підрозділ підприємства висококваліфікованими кадрами, що відповідають потребам розвитку бізнесу.

2. **Бачення проекту** (що з себе представляє система): розробити АС і створити програмний продукт, що автоматизує бізнес-процеси управління персоналом, що реалізовують виявлення кар'єрного потенціалу фахівців структурних підрозділів з урахуванням потреб розвитку бізнесу і мотивацій самих співробітників з метою задоволення кількісної і якісної потреби в персоналі.

3. **Комплекс задач** модуля:

0701 – Формування програми кар'єрного зростання персоналу підприємства на рік.

0702 – Аналіз рівня готовності конкретних співробітників до кар'єрного зростання.

703 – Формування кар'єрного плану співробітника.

4. Завдання на функціональність комплексу задач модуля, що реалізовується.

Задача 0701: На основі інформації, що зберігається в єдиній БД, про укомплектованість штатного розкладу виявити вакантні і умовно-вакантні посади в підрозділах підприємства з урахуванням процесів розвитку бізнесу. На основі інформації БД особистих карток спів робітників і виявлених вакантних і умовно-вакантних посад сформувати в електронному вигляді списки співробітників, яким доцільно взяти участь в програмі кар'єрного зростання з орієнтацією на підвищення займаної

посади. Списки співробітників повинні відбивати потреби розвитку бізнесу і побажання співробітників.

Задача 0702: Реалізувати безпаперову технологію анкетування шляхом створення електронної форми анкети, що містить сукупність компетенцій співробітника, що чинять вплив на ефективність його роботи на підвищуваній посаді. У анкеті передбачити три зони для оцінювання компетенцій співробітника по 5-бальній системі: його керівником, його підлеглими і самим співробітником. Для передачі електронної форми анкети використовувати технологію СЕД. Анкета формується на кожного із співробітників, які включені в список для участі в програмі кар'єрного зростання, сформований при рішенні задачі 0701. На основі проставлених оцінок розрахувати інтегральний показник рівня готовності співробітника до кар'єрного зростання і сформувати документ "Кар'єрна програма співробітника".

Задача 0703: На основі кар'єрної програми співробітника, сформованої в задачі 0702, визначити період (рік), підрозділ, посаду, яку планується зайняти співробітником. Сформувати кар'єрний план співробітника.

5. Визначити зміни в менеджменті персоналу і в організаційній роботі, які спричинять впровадження комплексу завдань модуля.

Розробив студент П.І.Б.
Керівник проекту посада, П.І.Б.

Додаток Д

Зразок оформлення реферату

РЕФЕРАТ

Пояснювальна записка: 75 сторінок, 30 рисунків, 20 таблиць, 12 додатків, 35 джерел.

Об’єкт дослідження – процеси відділу маркетингу ТОВ "НІКС СОЛЮШЕНС".

Мета розробки – створення проекту автоматизованого модуля "Оцінка ефективності рекламних заходів на підприємстві".

Поняття ефективності у рекламі, з одного боку, тісно пов’язано з поняттям економічної ефективності, з іншого – має свою власну специфіку.

У сучасній практиці маркетингу існує два основних підходи до оцінки ефективності реклами. У першому випадку реклама розглядається як інструмент маркетингової діяльності фірми і її економічний ефект виражається безпосередньо в обсягах продажів. У другому – розглядається комунікативна функція реклами, тобто значення того, наскільки докладно і якісно рекламно-інформаційний матеріал передає цільовій аудиторії відомості про товари, що рекламуються, чи формує бажану для рекламодавця реакцію потенційного споживача.

Тому у проекті обрано комплексний показник, який характеризує загальний рівень ефективності реклами і залежить як від економічної ефективності, так і від її комунікативного впливу на споживача. В основі оцінки економічної ефективності лежить функція прогнозування обсягу продажу для конкретного товару на планований період. Це дозволяє визначити позитивний ефект від застосування рекламних заходів. Здійснюється попередня оцінка рекламних заходів. Після проведення рекламних заходів відбувається постоцінка ефективності реклами, метою якої є перевірка прогнозного значення ефективності. Результати постоцінки будуть використовуватися для підведення підсумків рекламних заходів та прийняття управлінських рішень у майбутньому.

В цьому полягає новизна проектних розробок.

Пояснювальна записка комплексного курсового проекту містить результати розробки автоматизованого модуля "Оцінка ефективності

рекламних заходів на підприємстві". Проведено аналіз предметної області, розроблені моделі бізнес-процесів та інформаційних потоків модуля з використанням CASE-засобу розробки інформаційних систем BP Win. Розроблені специфікації бізнес-вимог, функціональних вимог до комплексу задач модуля з використанням UML-діаграм CASE-засобу Rational Rose. Спроектовані логічна та фізична моделі БД комплексу задач модуля з використанням CASE-засобу ER Win.

Розроблено постановку комплексу задач для вирішення на ПК та додаток за допомогою програмного засобу Microsoft Visual Studio 2008. Наведені результати тестування програмного продукту.

Результати розробки можуть бути впроваджені на підприємствах різної сфери діяльності.

РЕКЛАМА, РЕКЛАМНІ ЗАХОДИ, РЕКЛАМНИЙ БЮДЖЕТ, ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ, КОМУНІКАТИВНА ЕФЕКТИВНІСТЬ, МАРКЕТИНГ, ТЕХНОЛОГІЯ "КЛІЄНТ-СЕРВЕР", CASE-ДІАГРАМИ, БАЗА ДАНИХ, ПРЕТЕСТУВАННЯ, ПОСТТЕСТУВАННЯ.

Додаток Е

Зразок оформлення
титульного аркуша комплексного
курсового проекту

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Кафедра інформаційних
систем

КОМПЛЕКСНИЙ КУРСОВИЙ ПРОЕКТ
за темою_____<назва теми>_____

Виконав:
студент___курсу___гр.
_____
(підпис) (П.І.Б.)

Перевірив:
_____
(підпис) (посада, П.І.Б.)

Харків, 2010

Додаток Ж

Приклад тест-плану, сполученого зі звітом про проведення тестування

Група тестів: Робота з обліковими записами

Тестовий приклад: № 1

Призначення: перевірка того, що перед початком передачі даних перевіряється обліковий запис користувача й у випадку введення імені користувача за замовчуванням при знаходженні системи в стані "Максимальний захист" передача даних не відбувається.

Тест-вимоги, що перевіряються: 1.1, 1.2

Передумови для тесту: система повинна бути переведена в стан "Максимальний захист", а її настройки повинні мати значення за замовчуванням.

Критерій проходження тесту: всі реальні значення збігаються з очікуваними.

Таблиця Ж1

№ з/п	Крок сценарію	Очікуваний результат	Отриманий результат	Відмітка про проходження кроку сценарію (Так/Ні)
1	2	3	4	5
1.	Запустити термінальний клієнт і ввести IP-адресу 127.0.0.1	Повинне з'явитися запрошення: **TRANSFER>**	Запрошення **TRANSFER>** з'являється	Так
2.	Запустити процес передачі даних за допомогою введення команди **SEND DATA**	Повинне з'явитися запрошення: **Enter your credentials... Login:**	Запрошення **Enter your credentials... Login:** з'являється	Так

1	2	3	4	5
3.	Увести ім'я користувача **default**	Повинне з'явитися повідомлення: **Password:**	Повідомлення **Password:** з'являється	Так
4.	Увести пароль **default**	Повинне з'явитися повідомлення: **Default user blocked - system set to High security.**	З'являється повідомлення: **Your credetentials accepted. Data transfer started.**	Ні

Відмітка про проходження тесту (пройдений/не пройдений): не пройдений

Тестовий приклад: № 2

……

Тестовий приклад: № 3

……

Тестових прикладів виконано: 3
Тестових прикладів пройдено: 1

НАВЧАЛЬНЕ ВИДАННЯ

Методичні рекомендації до виконання комплексного курсового проекту

для студентів спеціальності "Інформаційні управляючі системи та технології" всіх форм навчання

Укладачі: **Золотарьова Ірина Олександрівна**
Мінухін Сергій Володимирович
Парфьонов Юрій Едуардович
Бутова Раїса Костянтинівна
Свердло Тамара Олексіївна

Відповідальний за випуск **Пономаренко В. С.**

Редактор **Байдак В. В.**

Коректор **Мартовицька-Максимова В. А.**

План 2010 р. Поз. № 207.
Підп. до друку Формат 60 х 90 1/16. Папір MultiCopy. Друк Riso.
Ум.-друк. арк. 5,0. Обл.-вид. арк. 6,25. Тираж прим. Зам. №

Видавець і виготівник — видавництво ХНЕУ, 61001, м. Харків, пр. Леніна, 9а

Свідоцтво про внесення до Державного реєстру суб'єктів видавничої справи ***Дк № 481 від 13.06.2001 р.***

Методичні рекомендації до виконання комплексного курсового проекту

для студентів спеціальності "Інформаційні управляючі системи та технології" всіх форм навчання