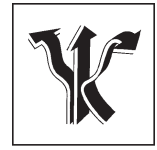
МІЖРЕГІОНАЛЬНА
АКАДЕМІЯ УПРАВЛІННЯ ПЕРСОНАЛОМ



МАУП

МЕТОДИЧНІ МАТЕРІАЛИ ЩОДО ЗАБЕЗПЕЧЕННЯ САМОСТІЙНОЇ РОБОТИ СТУДЕНТІВ

з дисципліни

“ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНИХ СИСТЕМ”

(для спеціалістів)

Київ
ДП «Видавничий дім «Персонал»
2008

Підготовлено доцентом кафедри інформатики та інформаційних технологій
Л. О. Левченко

Затверджено на засіданні кафедри інформатики та інформаційних технологій (протокол № 15 від 18.10.07)

Схвалено Вченою радою Міжрегіональної Академії управління персоналом

Левченко Л. О. Методичні матеріали щодо забезпечення самостійної роботи студентів з дисциплін "Програмне забезпечення автоматизованих систем" (для спеціалістів). — К.: ДП «Вид. дім «Персонал», 2008. — 36 с.

Методичні матеріали містять пояснювальну записку, зміст самостійної роботи з дисципліни "Програмне забезпечення автоматизованих систем", методичні вказівки до підготовки, написання та захисту реферату, а також список літератури

ПОЯСНЮВАЛЬНА ЗАПИСКА

Програмне забезпечення автоматизованих систем — центральна дисципліна циклу професійної та практичної підготовки спеціалістів з напряму "Комп'ютерні науки" зі спеціальності "Програмне забезпечення автоматизованих систем". Розроблення програмного забезпечення — одна з пріоритетних галузей світової економіки. Потреба контролювати процес розробки програмного забезпечення, прогнозувати та гарантувати вартість розробки, терміни та якість результатів привела до появи сукупності інженерних методів та засобів створення розроблення програмного забезпечення під загальною назвою "програмна інженерія". Сучасні великі проекти зі створення автоматизованих систем мають такі особливості:

- складність опису (достатньо велика кількість функцій, процесів, елементів даних та складні зв'язки між ними), яка потребує ретельного моделювання та аналізу даних і процесів;
- наявність сукупності тісно взаємодіючих компонентів (підсистем), які мають локальні завдання та цілі функціонування;
- відсутність повних аналогів, що обмежують можливість використання будь-яких типових проектних рішень та прикладних систем;
- функціонування в неоднорідному середовищі на кількох апаратних платформах;
- роз'єднаність та різнорідність окремих груп розробників за рівнем кваліфікації і традицій, що склалися, стосовно використання інструментальних засобів;
- значний термін виконання проекту, який обумовлений обмеженими можливостями колективу розробників, масштабами організації — замовника, різною готовністюі окремих підрозділів організації — замовника до запровадження автоматизованої системи.

Для успішної реалізації проекту об'єкт проектування має бути адекватно описаний, тобто повинні бути побудовані повні та несуперечливі моделі архітектури програмного забезпечення, що зумовлюють сукупність структурних елементів системи та зв'язки між ними, поведінку елементів системи в процесі їх взаємодії, а також ієрархію підсистем, що поєднують структурні елементи.

Моделі є засобами для візуалізації, опису, проектування та документування архітектури системи. Моделювання — центральна ланка

всієї діяльності зі створення якісного програмного забезпечення. Моделі будуються для розуміння та осмислення структури та поведінки майбутньої системи, полегшення управління процесом її створення, зменшення можливого ризику та документування прийнятих проектних рішень.

Розроблення точних моделей гарантує коректність архітектури програмного забезпечення. Оскільки складність систем постійно зростає, важливо мати ефективні методи моделювання. Хоча є багато й інших факторів, що впливають на успіх розроблення проекту, проте наявність суворого стандарту мови моделювання вельми важливе.

Мова моделювання повинна складатися з: елементів моделі — функціональних концепцій моделювання та їх семантики; нотації — візуального представлення елементів моделювання; документації з використання — правил застосування елементів у межах побудови тих чи інших типів моделей програмного забезпечення.

Оскільки розробка програм — це процес творчий, то необхідно мати інструмент, який сприяв би зменшенню частки ручної праці при програмуванні та надав великі можливості для творчості, вивільняючи програміста від виконання рутинних операцій та від помилок при прийнятті рішень.

Сучасні методи розробки програмного забезпечення — це структурні методи, що ґрунтуються на суворих формалізованих методах опису програмного забезпечення, та об'єктно-орієнтовані методи щодо прийняття технічних рішень.

Принципова різниця між структурним та об'єктно-орієнтованим підходом у способі декомпозиції системи.

Сутність структурного підходу полягає в наступному: система розбивається на функціональні підсистеми, функціональні підсистеми поділяються на підфункції, підфункції — на задачі, задачі — на конкретні процедури. При цьому система, що автоматизується, зберігає цілісне уявлення, в якому всі складові компоненти взаємопов'язані. Тобто розроблення системи "зверху — вниз".

Таким чином, при структурному підході в основу розроблення системи покладено принцип функціональної декомпозиції, при якій структура системи описується у термінах ієрархії її функцій та передачі інформації між окремими функціональними елементами.

При розробленні системи "знизу — вверх", тобто від окремих задач до розроблення усієї системи, її цілісність втрачається і виникають проблеми при опису інформаційної взаємодії окремих компонентів.

Об'єктно-орієнтований підхід використовує об'єктну декомпозицію, при цьому статична структура системи описується у термінах об'єктів та зв'язків між ними, а поведінка системи — в термінах обміну повідомленнями між об'єктами. Кожний об'єкт системи має власну поведінку і моделює поведінку об'єкта реального середовища. Об'єктно-орієнтована система будується на поняттях об'єкт і клас та з урахуванням їх еволюції. Важливою особливістю об'єктного підходу є узгодженість моделей діяльності організації та моделей системи, що проектується, від стадії формування вимог до стадії реалізації. Вимога узгодженості моделей виконується завдяки застосуванню абстрагування, модульності, поліморфізму на всіх стадіях розробки.

Головний недолік структурного підходу полягає в наступному: процеси та дані існують окремо один від одного (як у моделі діяльності організації, так і у моделі системи, що програмується), проектування ведеться від процесів до даних. Крім функціональної декомпозиції є структура даних, яка знаходиться на другому плані. В об'єктно-орієнтованому підході головна категорія об'єктної моделі — клас на елементарному рівні поєднує у собі як дані, так і операції, які над ними виконуються (методи). Головну перевагу об'єктно-орієнтованого підходу сформулював Граді Буч: об'єктно-орієнтовані системи є відкритими і значно легше піддаються внесенню змін, оскільки їх конструкція базується на стійких формах. Це дає можливість розвиватися системі поступово і не призводити до повної її переробки навіть у разі істотних змін початкових вимог. Об'єктна декомпозиція дає можливість створювати програмні системи меншого розміру шляхом використання загальних механізмів, що забезпечують економію виразних засобів. Використання об'єктного підходу істотно підвищує рівень уніфікації розробки та придатність для повторного використання не тільки програм, а й проектів, що врешті-решт приводить до створення середовища розробки та переходу до складального створення програмного забезпечення. Системи значно компактніші, що означає не лише зменшення обсягу програмного коду, а й здешевлення проекту за рахунок використання попередніх розробок. Об'єктна декомпозиція зменшує ризик створення складних систем програмного забезпечення, оскільки вона припускає еволюційний шлях розвитку системи на базі відносно невеликих підсистем. Об'єктна модель цілком природна, оскільки, насамперед, орієнтована на людське сприйняття світу, а не на комп'ютерну реалізацію.

Ця модель дає змогу повною мірою використати виразні можливості об'єктних та об'єктно-орієнтованих мов програмування.

До недоліків об'єктно-орієнтованого підходу належить деяке зниження продуктивності функціонування програмного забезпечення та високі початкові витрати.

Об'єктно-орієнтований підхід не дає негайної віддачі. Ефект від його застосування починає позначатися після розробки двох-трьох проектів і накопичення повторно використовуваних компонентів, що відображають типові проектні рішення у цій галузі.

В об'єктно-орієнтованому підході при створенні програм використовуються такі стилі програмування:

- процедурно-орієнтований (спрямований на представлення програми як множини процедур, що викликаються почергово);
- об'єктно-орієнтований (спрямований на представлення програми як набору взаємодіючих об'єктів);
- логіко-орієнтований (спрямований на виконання цілей, виражених у термінах счислення предикатів);
- орієнтований на правила (спрямований на виконання правил "якщо, то");
- орієнтований на обмеження.

При використанні об'єктно-орієнтованого стилю програміст створює програмні об'єкти та наділяє їх певною поведінкою, реакцією на зміни зовнішніх умов. Такі об'єкти взаємодіють між собою, виконують певні задачі, приймають, обробляють та передають дані.

З об'єктно-орієнтованим програмуванням тісно пов'язане об'єктно-орієнтоване проектування, яке ще на етапі задуму системи розглядає її та аналізує як набір взаємодіючих об'єктів. При об'єктно-орієнтованому стилі програмування концептуальною базою виступає об'єктна модель, яка має чотири головних властивості:

- абстрагування — виокремлення істотних характеристик об'єкта, що відрізняють його від інших видів об'єктів;
- інкапсуляція — приховування внутрішньої реалізації об'єкта за інтерфейсом, який представляє цей об'єкт;
- модульність — здатність системи бути представленою у вигляді сильно або слабо пов'язаних між собою модулів;
- ієрархія — упорядкування абстракцій та розташування їх за рівнями.

За відсутності будь-якої з цих властивостей модель не може бути об'єктно-орієнтованою. Існують також три додаткові властивості,

які є корисними в об'єктній моделі, проте вони не є обов'язковими. Це такі властивості:

- типізація — створення об'єктів на основі шаблонів певного типу;
- паралелізм — здатність системи оброблювати кілька повідомлень або задач паралельно;
- збереженість — здатність зберігати не тільки дані, а й об'єкти у проміжках між окремими запусками системи.

Сучасні методи розробки програмного забезпечення ґрунтуються на використанні наочних графічних моделей: для опису архітектури програмного забезпечення в різних аспектах (як статичної структури, так і динамічної структури) використовуються схеми та діаграми. Наочність та суворість засобів структурного та об'єктно-орієнтованого аналізу дає змогу розробникам та майбутнім користувачам системи із самого початку брати участь у створенні, обговоренні та розумінні основних технічних рішень. Мовою моделювання в більшості існуючих методах об'єктно-орієнтованого аналізу та проектування є уніфікована мова моделювання — UML (Unified Modeling Language), прийнята у 1997 р. групою промислових стандартів OMG (Object Management Group) як стандартна мова моделювання в галузі об'єктно-орієнтованих методів і технологій. Усі великі виробники програмного забезпечення: Microsoft, IBM, Hewlett-Packard, Oracle, Sybase використовують цю мову у своїх розробках.

Розробники програмного забезпечення широко послуговуються програмно-технологічними засобами спеціального класу — CASE-засобами, що реалізують CASE-технологію, тобто автоматизовану технологію створення та супроводження програмного забезпечення.

CASE-технологія становить собою сукупність методів проектування автоматизованих систем, набір інструментальних засобів, що дають змогу наочно моделювати предметну область, аналізувати цю модель на різних стадіях розробки та супроводження, розробляти застосування відповідно до інформаційних потреб користувачів. Більшість існуючих CASE-засобів ґрунтуються на засобах структурного або об'єктно-орієнтованого аналізу та проектування, використовуючи специфікації у вигляді діаграм або текстів опису зовнішніх вимог, зв'язків між моделями системи, динаміки поведінки системи та архітектури програмних засобів.

У структурному підході використовуються дві групи засобів, що описують функціональну структуру системи та відносини між даними. Кожній групі засобів відповідають певні види моделей (діаграм). До першої групи належать такі методи:

- функціональне моделювання (метод SADT — Structured Analysis and Design Technique, метод структурного аналізу та проектування), функціональні діаграми (IDEF0);
- моделювання потоків даних (DFD — Data Flow Diagrams), діаграми потоків даних.

До другої групи належать моделювання даних з використанням підходу "сутність — зв'язок" (ERD — Entity-Relationship Diagrams, діаграми "сутність — зв'язок").

Методи та інструментальні засоби проектування (CASE-засоби) становлять центральну частину формалізованої складової виконання будь-якого проекту зі створення програмного забезпечення. Метод програмного забезпечення є організованою сукупністю процесів створення низки моделей, які описують різні аспекти системи, що розробляється, з використанням чітко визначеної нотації.

На більш формальному рівні метод визначається як сукупність наступних складових:

- концепцій і теоретичних основ. Основою може бути структурний або об'єктно-орієнтований підхід;
- нотацій, що використовуються для побудови моделей статистичної структури та динаміки поведінки системи, що проектується. Як нотації зазвичай використовують графічні діаграми (діаграми потоків даних та діаграми "сутність — зв'язок" для структурного підходу; діаграми варіантів використання, діаграми класів — для об'єктно-орієнтованого підходу);
- процедури, що визначають практичне застосування методу (послідовність та правила побудови моделей, критерії, що використовуються для оцінювання результатів).

Методи реалізуються через конкретні технології та методики, що їх підтримують, стандарти та інструментальні засоби, які забезпечують виконання процесів життєвого циклу програмного забезпечення.

Технологія проектування програмного забезпечення визначається як сукупність технологічних операцій проектування в їх послідовності та взаємозв'язках, що призводить до розроблення проекту програмного забезпечення.

Реальне застосування будь-якої технології проектування програмного забезпечення автоматизованої системи в конкретній організації та конкретному проекті неможливе без вироблення низки стандартів (правил, узгоджень), яких можна дотримуватися усі учасники проекту. До таких стандартів належать:

- стандарт проектування;
- стандарт оформлення програмної документації;
- стандарт інтерфейсу кінцевого користувача із системою.

Стандарт проектування встановлює:

- набір необхідних моделей (діаграм) на кожній стадії проектування та ступінь їх деталізації;
- правила фіксації проектних рішень на діаграмах;
- набір атрибутів для об'єктів;
- правила оформлення діаграм;
- вимоги до конфігурації робочих місць розробників;
- механізм забезпечення спільної роботи над проектом.

Стандарт оформлення документації визначає:

- комплектність, склад та структуру документації на кожній стадії проектування;
- вимоги до оформлення документації;
- правила підготовки, розгляду, узгодження та затвердження документації із зазначенням граничних термінів для кожної стадії;
- вимоги до налаштування видавничої системи, яка використовується як вбудований засіб підготовки документації;
- вимоги до налаштування CASE-засобів для забезпечення підготовки документації відповідно до встановлених правил.

Стандарт інтерфейсу кінцевого користувача із системою регламентує:

- правила оформлення екранів (шрифти, кольорова палітра), склад і розташування вікон та елементів управління;
- правила оформлення текстів підказок;
- перелік стандартних повідомлень.

Основу курсу становлять програмні засоби для проведення проектування складних систем та виконання проектних процедур, що використовують типові системи автоматизованого проектування систем такого типу.

Обмежена кількість лекційних і практичних занять має компенсуватися активною самостійною роботою студентів. Це вимагає сис-

тематичної копіткої праці над навчально-методичною та науковою літературою, законодавчими і нормативними актами, виділення додаткового часу для оволодіння навичками застосування на практиці вивченого теоретичного матеріалу. Метою самостійної роботи є не лише глибоке й повне засвоєння знань, а й розвиток індивідуальних здібностей студентів, їхніх умінь опановувати типові системи автоматизованого проектування систем, використовуючи власні сили і творчий підхід.

Теоретичний матеріал з автоматизованого проектування систем потребує багаторазового підкріплення великої кількості практичних прикладів у різних сферах життя. Студенти мають здобути навички самостійного виконання усіх етапів розробки програмного забезпечення (вивчення предметної області, визначення об'єктів проектування, знання технології та процедур проектування, розроблення програмного забезпечення, відладка, тестування, впровадження пілотного варіанта, дослідна експлуатація, супроводження, модернізація). Це вимагає від студента систематичного виконання практичних завдань протягом семестру та підготовки до кожного практичного заняття.

Самостійна робота має такі складові і форми їх оцінювання:

- підготовка та аудиторна робота під час практичних і лабораторних занять. Результати її оцінюються під час поточного контролю;
- виконання самостійних робіт у формі створення автоматизованих систем, систем управління базами даних у різних предметних сферах, складання звітів на електронних або паперових носіях;
- опрацювання програмного матеріалу зі змістового модуля та оцінювання її результатів під час проміжного контролю;
- виконання письмової контрольної роботи або тестування;
- звіт про проходження практики;
- звіт про науково-дослідну роботу, результати якої можуть бути використані при написанні випускної роботи.

Мета вивчення дисципліни — формувати у студентів необхідний рівень знань у сфері програмної інженерії, а саме: вивчення мови об'єктно-орієнтованого моделювання UML, CASE-засобів, які підтримують як структурний (Silverrun, Oracle Designer, Erwin, BPwin), так і об'єктно-орієнтований підходи (Rational Rose) проектування

програмних засобів складних систем та практичне уміння виконання типових автоматизованих проектних процедур.

У результаті самостійного вивчення дисципліни студенти повинні:

- знати основні поняття (програмна інженерія, програмне забезпечення, життєвий цикл програмного забезпечення, процеси життєвого циклу, модель життєвого циклу програмного забезпечення, стадії життєвого циклу, каскадна модель, спіральна модель);
- знати методи структурного аналізу та проектування;
- знати сутність об'єктно-орієнтованого підходу до проектування програмного забезпечення;
- знати шаблони проектування в галузі об'єктно-орієнтованого аналізу та проектування;
- отримати навички розробки ефективних програм або модифікувати існуючі;
- вміти створювати модель архітектури програмного забезпечення;
- знати організаційну структуру об'єкта у відповідній сфері;
- вміти розробляти інформаційну модель конкретної системи;
- вміти розробляти функціональну модель системи загалом та її підсистем;
- знати складові програмно-технічної платформи (апаратні засоби, операційні системи, СУБД, мови програмування, що використовуються при проектуванні автоматизованих систем);
- знати типові процедури автоматизованого проектування;
- знати типи та структуру файлів, які необхідно формувати в системі;
- вміти працювати на відповідних АРМ-х автоматизованої системи;
- знати засоби інформаційної безпеки;
- вміти розробляти документацію на проект.

ЗМІСТ САМОСТІЙНОЇ РОБОТИ з дисципліни "ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНИХ СИСТЕМ"

Тема 1. "Життєвий цикл" складної системи та модельне його уявлення

1. Міжнародний стандарт ISO / IEC 12207 1995 "Information Techology — Software Life Cycle Process".
2. Моделі та стадії життєвого циклу програмного забезпечення.
3. Характеристика RAD-підходу (Rapid Application Development) до розроблення моделі життєвого циклу.
4. Метод та технологія проектування програмного забезпечення.
5. Вимоги до технології проектування програмного забезпечення.
6. Стандарти розроблення проекту.

Література [1; 3; 4; 11; 15]

Теми рефератів

1. Проектування економічних інформаційних систем.

Література [11; 15]

2. Регіональні бази даних.

Література [32]

3. Система екологічного моніторингу.

Література [32]

4. Стандарти інформаційних технологій.

Література [28; 29]

Питання для самоконтролю та співбесіди

1. Як відповідно до стандарту ISO / IEC 12207 визначається програмне забезпечення, процес?
2. Що таке модель життєвого циклу програмного забезпечення?
3. З яких стадій складається життєвий цикл програмного забезпечення?
4. Які моделі життєвого циклу найпоширеніші?
5. У чому полягають переваги й недоліки спіральної моделі?
6. Які принципові особливості каскадної моделі?
7. Охарактеризуйте три складові RAD-підходу при розробленні життєвого циклу програмного забезпечення.
8. Що таке метод проектування програмного забезпечення?

9. Що таке технологія програмного забезпечення?
10. Які вимоги пред'являються до технології проектування програмного забезпечення?
11. Які необхідні стандарти для виконання конкретного проекту?
12. Що таке стандарт проектування?
13. Схарактеризуйте стандарт оформлення програмної документації.
14. Схарактеризуйте стандарт інтерфейсу кінцевого користувача із системою.

Тестові завдання

1. Усі процеси життєвого циклу програмного забезпечення поділяються на групи:

A. основні процеси;
B. допоміжні процеси;
C. організаційні процеси;
D. інформаційні процеси;
E. процеси документування.

2. Стадія формування вимог до програмного забезпечення складається з наступних етапів:

A. планування робіт;
B. проведення попереднього обстеження діяльності об'єкта, який автоматизується;
C. побудови двох видів моделей діяльності організації ("AS-IS", "TO-BE");
D. вибору програмних засобів;
E. проектування архітектури системи.

3. Найпоширенішими моделями життєвого циклу програмного забезпечення є:

A. ієрархічна модель;
B. спіральна модель;
C. функціональна модель;
D. каскадна модель;
E. інформаційна модель.

4. Відповідно до RAD-підходу життєвий цикл програмного забезпечення включає стадії:

A. аналізу та планування вимог;

B. проектування;
C. реалізації;
D. тестування;
E. впровадження;
F. розроблення документації.

5. Метод проектування програмного забезпечення становить:

A. сукупність концепцій та теоретичних основ;
B. сукупність інструментальних засобів;
C. сукупність нотацій;
D. сукупність процедур;
E. сукупність підходів до побудови моделі;
F. сукупність інформаційних потоків.

6. Стандарт проектування встановлює:

A. набір діаграм кожної стадії проектування;
B. набір об'єктів та їх атрибутів;
C. вибір операційної системи та інструментальних засобів проектування;
D. правила спільної роботи над проектом.

7. При розробленні RAD-проектів під функціональною точкою розуміють один з елементів системи:

A. вхідний документ або екранну форму;
B. звіт, документ;
C. сукупність записів;
D. запит;
E. таблицю;
F. поле.

8. RAD-проект виконується терміном до:

A. року;
B. півроку;
C. трьох місяців;
D. місяця.

9. Вимоги до технології проектування програмного забезпечення повинні:

A. відповідати стандарту ISO — IEC 12207;

B. забезпечити декомпозицію проекту на частини;
C. мінімізувати час для отримання працездатного програмного забезпечення;
D. забезпечити розроблення системи в межах виділеного бюджету, із заданою якістю, у визначений термін.

10. Особливістю каскадного підходу є:

A. перехід на наступну стадію здійснюється тільки по завершенні попередньої стадії проектування;
B. можливе повернення на попередні стадії проектування;
C. результат роботи однієї стадії є вхідними даними для наступної стадії;
D. відсутня можливість повернення на попередню стадію.

Тема 2. Структурний підхід до проектування програмного забезпечення

1. Сутність структурного підходу до проектування програмного забезпечення.
2. Метод функціонального моделювання SADT (Structured Analysis and Design Technique).
3. Моделювання потоків даних (процесів) — діаграми потоків даних (DFD — Data Flow Diagrams).
4. Моделювання даних засобами "сутність — зв'язок" (ERD — Entity-Relationship Diagrams).
5. Метод IDEF1X.

Література [1; 3; 6–8; 11; 15]

Теми рефератів

1. Стандарт IDEF0 — федеральний стандарт США.
   Література [Інтернет-ресурс http://www.idef.com; 8; 11]
2. Сучасні засоби проектування інформаційних систем.
   Література [10–13, Інтернет-ресурси]
3. Метод моделювання даних Баркера.
   Література [6; 7]

Питання для самоконтролю та співбесіди

1. У чому полягає сутність структурного підходу?
2. Що таке метод SADT?
3. Які основні елементи методу SADT?

4. З чого складається функціональна модель SADT?
5. Які розрізняють типи зв'язків при моделюванні бізнес-процесів за допомогою методу SADT?
6. Що таке стандарт IDEF0?
7. У чому полягає сутність моделювання потоків даних?
8. Які основні компоненти діаграм потоків даних?
9. У чому полягає моделювання даних за допомогою методу ERD?
10. Які базові поняття методу ERD?
11. У чому полягає сутність методу IDEF1X?

Тестові завдання

1. Декомпозиція складної програмної системи означає наступне:

A. кількість зв'язків між окремими підсистемами має бути максимальною;
B. зв'язність окремих частин;
C. кількість зв'язків між окремими підсистемами має бути мінімальною;
D. кожна підсистема повинна інкапсулювати (приховувати) свій вміст від інших підсистем;
E. кожна підсистема має бути відкритою для інших підсистем;
F. кожна підсистема повинна мати чітко визначений інтерфейс з іншим підсистемами.

2. У програмній інженерії існують такі підходи щодо розроблення програмного забезпечення:

A. ієрархічний, тобто визначаються рівні підпорядкованості у системі та виділяються блоки з можливістю додавання нових;
B. функціонально-модульний (структурний), тобто структура системи описується у термінах ієрархії її функцій та передачі інформації між окремими функціональними компонентами;
C. об'єктно-орієнтований, тобто структура системи описується в термінах об'єктів та зв'язків між ними, поведінка системи описується в термінах обміну повідомленнями між об'єктами;
D. модульний підхід, тобто система є відкритою і будується за принципами модульного програмування.

3. Метод SADT – це:

A. ієрархія функціональних компонентів (процесів), пов'язаних потоками даних;

B. сукупність правил та процедур, призначених для побудови функціональної моделі об'єкта будь-якої предметної області;
C. сукупність діаграм, які моделюють дані та їх відношення.

4. Метод SADT базується на:
A. перетворенні вхідних даних у вихідні;
B. графічному представленні блочного моделювання (блок відображає функцію, дуга — інтерфейс входу-виходу у блок);
C. обмеженні кількості блоків на кожному рівні (3–6 блоків);
D. зв'язності діаграм (номера блоків);
E. відсутності імен, що повторюються;
F. розділу входів та управлінь (правило визначення ролі даних);
G. відсутність впливу організаційної структури на функціональну модель;
H. джерелах інформації, які породжують інформаційні потоки.

5. Стандарт IDEF0 (Icam DEFinition) — це:
A. підмножина SADT;
B. метод побудови моделі, який є еквівалентом реляційної моделі у третій нормальній формі;
C. метод розроблення концептуальної моделі даних.

6. При моделюванні бізнес-процесів з використанням методу SADT розрізняють такі типи зв'язків:
A. функціональні, послідовні, процедурні;
B. випадкові, логічні, тимчасові, процедурні, комунікаційні, послідовні, функціональні;
C. тимчасові, логічні, функціональні;
D. логічні, випадкові, комунікаційні.

7. Відповідно до діаграм потоків даних (DFD) модель системи визначається як:
A. ієрархія діаграм потоків даних, що описує асинхронний процес перетворення інформації від її введення у систему до видачі користувачу;
B. декомпозиція діаграм процесів, коли процес стає елементарним і далі деталізації не підлягає;
C. сукупність вимог до системи, що проектується.

8. Основними компонентами діаграм потоків даних є:

A. системи та підсистеми, процеси, потоки даних;
B. сутності, процеси, потоки даних;
C. зовнішні сутності, системи та підсистеми, процеси, накопичувачі даних, потоки даних;
D. сутності, процеси, накопичувачі даних.

9. Основними поняттями діаграми "сутність – зв'язок" є:

A. атрибут;
B. потік;
C. сутність;
D. зв'язок;
E. діаграма;
F. супертипи та підтипи;
G. рекурсивний зв'язок;
H. зв'язки, що взаємно виключаються;
I. непереміщувані зв'язки.

10. Обов'язковий атрибут у методі ERD позначається символом:

A. #;
B. ;
C. *;
D. ;&;
E. °.

11. Унікальний ключ в сутності позначається символом:

A. $;
B. #;
C. *;
D. •;
E. □.

12. Основними компонентами IDEF1X-діаграм є:

A. сутність;
B. потужність зв'язку;
C. зв'язок;
D. атрибути.

13. Атрибут первинного ключа в IDEF1X-діаграмах позначається:

A. символом #;
B. символом •;
C. відокремлюється в сутності горизонтальною лінією;
D. символом →.

Тема 3. Об'єктно-орієнтований підхід до проектування програмного забезпечення

1. Сутність об'єктно-орієнтованого підходу.
2. Уніфікована мова моделювання UML, загальна характеристика.
3. Діаграми варіантів використання, загальна характеристика.
4. Діаграми класів, загальні відомості.
5. Діаграми взаємодії, загальні відомості.

Література [13; 15; 21–24; 34]

Теми рефератів

1. Проектування складних програмних засобів для інформаційних систем.

Література [4; 13]

2. Об'єктно-орієнтовані технології розробки складних програмних систем.

Література [13; 31]

3. Автоматизоване проектування.

Література [5; 27]

Питання для самоконтролю та співбесіди

1. У чому полягає сутність об'єктно-орієнтованого підходу?
2. Які основні елементи об'єктної моделі?
3. Які основні поняття об'єктно-орієнтованого підходу?
4. Які переваги та недоліки має об'єктно-орієнтований підхід у порівнянні із структурним?
5. Що означає поняття абстрагування?
6. Поясність значення терміна "інкапсуляція"?
7. Що таке модульність?
8. Що таке типізація у об'єктно-орієнтованому підході і як вона використовується?
9. Схарактеризуйте поняття "паралелізм".
10. Що означає властивість об'єкта?

11. Що таке поліморфізм?
12. Що означає спадкоємність?
13. Як розподілити обов'язки між класами та об'єктами?
14. Як визначити функції, що виконують конкретні класи?
15. Що таке діаграми класів?
16. Що таке асоціації?
17. Що таке операції?
18. Що таке узагальнення та обмеження в діаграмах класів?
19. Що таке множинна та динамічна класифікація?
20. Що таке агрегація та композиція?
21. Для чого використовуються діаграми варіантів використання?
22. Для чого використовуються діаграми класів?
23. Для чого використовуються діаграми взаємодії і коли їх доцільно використовувати?
24. Що таке кооперативні діаграми?
25. Яке призначення діаграм стану?
26. Схарактеризуйте діаграми діяльностей.
27. Схарактеризуйте діаграми компонентів.
28. Схарактеризуйте діаграми розташування.

Тестові завдання

1. Основними елементами об'єктної моделі є:

A. декомпозиція, абстрагування, модульність, інкапсуляція;
B. ієрархія, типізація, паралелізм, стійкість;
C. абстрагування, модульність, інкапсуляція;
D. поліморфізм, спадкоємство;
E. A+B;
F. C+D;
G. B+C+D.

2. Основними поняттями об'єктно-орієнтованого підходу є:

A. задача;
B. операція;
C. клас;
D. структура;
E. об'єкт;
F. функціональність.

3. Поліморфізм — це:

A. обмеження, що накладається на клас об'єктів та перешкоджає взаємозамінюваності різних класів (або звужує її можливість);
B. властивість об'єкта існувати у часі або в просторі;
C. здатність об'єкта належати більш, ніж одному типу;
D. властивість об'єкта знаходитися в активному або пасивному стані.

4. Спадкоємність означає:

A. побудову нових класів на основі існуючих з можливістю додавання або перевизначення даних та методів;
B. упорядковану систему абстракцій, розташованих за рівнями;
C. процес відокремлення один від одного окремих елементів системи, що визначає їх устрій та поведінку;
D. виокремлення істотних характеристик деякого об'єкта.

5. Діаграма варіантів використання — це:

A. послідовність дій (транзакцій), що виконує система у відповідь на подію, яка ініціюється деяким зовнішнім об'єктом;
B. сукупність ролей, яку виконують користувачі стосовно системи;
C. перелік усіх подій, які відбуваються у зовнішньому світі, на які система певним чином має реагувати.

6. Для діаграми класів існують такі види статичних зв'язків:

A. атрибути;
B. операції;
C. асоціації;
D. обмеження;
E. підтипи.

7. Діаграми взаємодії використовуються для:

A. опису поведінки груп об'єктів, які взаємодіють;
B. відображення потоків передачі повідомлень між програмними об'єктами та викликів методів;
C. створення діаграм послідовності та корпоративних діаграм;
D. для опису поведінки одного об'єкта в багатьох варіантах використання.

8. Діаграмами стану є:

A. засіб опису поведінки системи;
B. засіб опису динаміки поведінки одного об'єкта;
C. засіб опису поведінки об'єктів, що взаємодіють;
D. опис процесу зміни стану об'єкта.

9. Діаграми діяльності доцільно використовувати для:

A. опису поведінки одного процесу;
B. опису поведінки послідовних процесів;
C. опису поведінки великої кількості паралельних процесів.

10. Діаграми компонентів призначені для:

A. аналізу моделі системи на фізичному рівні;
B. аналізу компонентів програмного забезпечення та зв'язків між ними;
C. використання виконуваних компонентів та бібліотек коду;
D. визначення програмних компонентів клієнта;
E. визначення програмних компонентів серверу.

Тема 4. CASE – засоби

1. Загальна характеристика CASE-засобів.
2. Класифікація CASE-засобів.
3. Технологія впровадження CASE-засобів.
4. Визначення потреб у CASE-засобах.
5. Критерії вибору та оцінювання CASE-засобів.
6. Пілотний проект.
7. Характеристика CASE-засобу SILVERRUN.
8. Характеристика CASE-засобу ORACLE DESIGNER.
9. Характеристика CASE-засобів ERrwin та BPwin.
10. Характеристика CASE-засобу RATIONAL ROSE.

Література [11–16; 21–24; 31; 51–55; Інтернет-ресурси]

Теми рефератів

1. Огляд промислових технологій проектування програмного забезпечення.

Література [5; 15; 27]

2. Електронна технологія DATARUN.

Література [15; 25; 26; 55]

3. Технологія RUP компанії Rational Software.

Література [14; 15; 16]

4. Метод розробки програмного забезпечення компанії ORACLE-CMD (Custom Development Method).

Література [15; 25; 26]

5. Метод управління проектом компанії ORACLE-PJM (Project Menegement Method).

Література [15; 25; 26]

6. Метод впровадження прикладного програмного забезпечення компанії ORACLE-AIM (Application Implement Method).

Література [15; 25; 26]

7. Реінженірінг бізнес-процесів компанії ORACLE-BPR (Business Process Reengineering).

Література [15; 25; 26]

8. Метод створення сховищ даних компанії ORACLE-DWM (Data Warehouse Method).

Література [15; 25; 26]

Питання для самоконтролю та співбесіди

1. Що таке CASE-засоби?
2. Яке призначення репозитарію?
3. Наведіть приклад класифікації CASE-засобів.
4. Які ви знаєте CASE-засоби, що належать до засобів аналізу та проектування?
5. Назвіть CASE-засоби, що належать до засобів проектування баз даних.
6. Назвіть CASE-засоби, що належать до засобів управління вимогами.
7. Назвіть CASE-засоби, що належать до засобів управління конфігурацією програмних забезпеченнь.
8. Назвіть CASE-засоби, що належать до засобів документування.
9. Назвіть CASE-засоби, що належать до засобів тестування.
10. Назвіть CASE-засоби, що належать до засобів управління проектом.
11. Назвіть CASE-засоби, що належать до засобів реверсного інжинірингa.
12. У чому полягає технологія впровадження CASE-засобів?
13. З яких етапів складається процес впровадження CASE-засобів?
14. Що отримує організація в результаті позитивного впровадження CASE-засобів?

15. Які цілі висуває організація при визначенні потреб щодо застосування CASE-технологій?
16. Наведіть перелік витрат на впровадження CASE-засобів.
17. Які основні критерії щодо вибору CASE-засобів?
18. Що таке пілотний проект?
19. З яких кроків складається пілотний проект?
20. Що дає впровадження пілотного проекту?
21. З яких модулів складається CASE-засіб SILVERRUN та які функції вони виконують?
22. Яка структура та функції CASE-засобу ORACLE DESIGNER?
23. Яка структура та функції CASE-засобів ERwin та BPwin?
24. Яка структура та функції CASE-засобу RATIONAL ROSE?

Тестові завдання

1. Для CASE-засобів властиві наступні особливості:

A. наявність потужних графічних засобів для опису та документування системи, які забезпечують інтерфейс з розробником;
B. наявність засобів тестування;
C. інтеграція окремих компонентів CASE-засобів, що забезпечують керованість процесом розробки програмного забезпечення;
D. використання сховища проектних метаданих — репозитарія;
E. наявність засобів управління проектом.

2. Репозитарій — це:

A. сховище версій проекту та його окремих компонентів;
B. графічні засоби аналізу та проектування;
C. засоби управління конфігурацією програмного забезпечення;
D. засоби розробки застосувань.

3. У CASE-засобах реалізуються такі види контролю:

A. контроль синтаксису діаграм та типів їх елементів;
B. контроль повноти та спроможності діаграм (усі елементи діаграм повинні бути ідентифіковані та відображатися у репозитарії);
C. наскрізний контроль діаграм одного або різних типів на предмет їх спроможності по рівням — вертикальне та горизонтальне балансування;
D. контроль метаданих;
E. контроль засобів тестування.

4. CASE-засоби поділяються за:

A. типами;
B. видами діаграм;
C. категоріями;
D. класами.

5. Процес впровадження CASE-засобів включає такі етапи:

A. вивчення предметної області;
B. визначення потреб;
C. проведення випробувань;
D. оцінку та вибір;
E. виконання пілотного проекту;
F. впровадження.

6. При впровадженні CASE-засобів організація витрачає кошти на наступні статті:

A. вибір та придбання CASE-засобів;
B. встановлення та налаштування;
C. підготовка документації, стандартів та процедур використання засобів;
D. опануваня засобів розробниками;
E. інтеграція з іншими засобами та даними.

7. Репозитарій для реалізації CASE-засобів забезпечує:

A. адміністрування (контроль та забезпечення цілісності проектних даних);
B. автоматичне резервування;
C. захист від несанкціонованого доступу;
D. аналіз відмов у критичних застосуваннях.

8. Основні вимоги щодо вибору CASE-засобів:

A. підтримка повного життєвого циклу програмного забезпечення із забезпеченням його еволюційного розвитку;
B. забезпечення цілісності проекту та контроль за його станом;
C. незалежність від програмно-апаратної платформи та СУБД;
D. підтримка одночасної роботи груп розробників;
E. розроблення застосувань “клієнт–сервер”;
F. відкрита архітектура, можливість експорта-імпорта.

9. Пілотний проект — це:

A. широкомасштабне використання CASE-засобів;
B. первинне використання CASE-засобів;
C. реальне використання CASE-засобів.

10. Проектний проект переслідує такі цілі:

A. придбання або відмову у придбанні CASE-засобів;
B. підтвердження вірогідності результатів оцінювання та вибору CASE-засобів;
C. надання інформації щодо практичного впровадження CASE-засобів;
D. набуття користувачем власного досвіду використання CASE-засобів;
E. придбання ліцензій та навчання вузького кола спеціалістів.

Тема 5. Rational Rose — CASE-технологія об'єктно-орієнтованого моделювання великих складних систем

1. Загальна характеристика Rational Rose.
2. Головне вікно та набір інструментів Rational Rose.
3. Компоненти Rational Rose.
4. Діаграми сценаріїв (Use case diagram).
5. Діаграми топології (Deployment diagram).
6. Діаграми стану (Statechart diagram).
7. Діаграми активності (Activity diagram).
8. Діаграми взаємодії (Interaction diagram).
9. Діаграми послідовностей дій (Sequence diagram).
10. Діаграми співпраці (Collaboration diagram).
11. Діаграми класів (Class diagram).
12. Діаграми компонентів (Component diagram).

Література [14; 16; 18]

Теми рефератів

1. Засіб документування Rational SoDA.
   Література [14; 18; Інтернет-ресурси]
2. Засіб управління вимогами Requisite Pro.
   Література [14; 18; Інтернет-ресурси]
3. Засоби тестування SQA Suite, Performance Studio.
   Література [14; 18; Інтернет-ресурси]

4. Засоби конфігурування ClearCase, PVCS.

Література [14; 18; Інтернет-ресурси]

5. Засіб проектування компанії Rational Software — Rational Real Time.

Література [14; 18; Інтернет-ресурси]

Питання для самоконтролю та співбесіди

1. Які переваги надає застосування CASE-засобу Rational Rose для проектувальника та розробника моделі складної системи?
2. Як налаштовується робочий стіл Rational Rose?
3. Яка стуруктура меню програми?
4. Як створити нову модель та її зберегти?
5. Як змінити формат діаграми, шрифт, колір?
6. Які є можливості зміни вигляду діаграм?
7. Схарактеризуйте діаграми сценаріїв Use Case.
8. Які значки знаходяться у рядку інструментів діаграми Use Case та яке їх призначення?
9. Які значки специфічні тільки для діаграми Use Case?
10. Як за допомогою діаграми створити сценарій поведінки?
11. Для чого призначені діаграми топологій Deployment?
12. Для чого призначений об'єкт Processor?
13. Які можливості контекстного меню для об'єкта Processor?
14. Які специфікації можна налаштувати для об'єкта Processor?
15. Яке призначення об'єкта Device?
16. Яке призначення об'єкта Connection?
17. Для чого призначені діаграми стану Statechart?
18. Які інструменти доступні в діаграмі Statechart?
19. Які бувають переходи між станами?
20. Які специфікації можна задати для переходів між станами?
21. Що таке історія станів?
22. Для чого призначена діаграма взаємодії Sequence?
23. Які види повідомлень дозволяє відобразити діаграма взаємодії?
24. Як налаштувати відображення часу життя об'єкта?
25. Як створити класи, не виходячи з діаграми?
26. Який порядок обміну повідомленнями може бути заданий?
27. Якою може бути задана частота обміну повідомленнями?
28. Яке призначення діаграми взаємодії Collaboration?
29. Як перенести дані між діаграмами Sequence та Collaboration?
30. Які інструменти доступні в діаграмі Collaboration?

31. Які команди доступні через контекстне меню об'єктів у діаграмі Collaboration?
32. Які є можливості налаштування області видимості об'єкта?
33. Які є можливості налаштування властивостей повідомлень?
34. Яке призначення діаграми компонентів — Component diagram?
35. Які інструменти надає Component diagram?
36. Які властивості можна встановити у компонентів?
37. Що таке стереотип та для чого його застосовують?
38. Які властивості можна налаштувати на вкладці COM діалогового вікна "Component Specification for EnvironmentalController"?
39. Яке призначення діаграми класів — Class diagram?
40. Якими способами можна створити діаграму класів?
41. Які інструменти доступні для діаграми класів?
42. Яке контекстне меню надає контекстне меню класу?
43. Як налаштувати властивості атрибутів класу?
44. Які види зв'язків доступні у діаграмі класів?
45. Які специфікації є доступними для зв'язку Unadirectional Association?
46. Для чого використовуються зв'язки Association Class?
47. Яке призначення зв'язків Dependency та Generalization?
48. Що таке Parameterized Class та як звичайний клас змінити на параметризований?
49. Як налаштувати властивості методів класу?
50. У чому полягає особливість застосування діаграм співпраці?
51. Як перемикатися між діаграмами?
52. Використовуючи пункт Export model (експорт моделі), які елементи моделі можна експортувати?
53. Які типи файлів можна імпортувати в Rational Rose?
54. Поясніть призначення механізму "Карти віртуальних шляхів".
55. Яка інформація відображається у рядку стану?
56. Які звіти доступні та для чого вони використовуються?
57. Яка команда встановлює додаткові модулі, завдяки яким можна організувати роботу як з продуктами компанії Rational, так і продуктами інших виробників?
58. Що таке Add-In-менеджер та для чого він використовується?
59. Для чого використовуються пункти меню Tools, WinDows, Help?

Тестові завдання

1. При розробленні нової моделі на робочому столі відкривається:

A. діаграма сценаріїв;
B. діаграма топології;
C. діаграма класів;
D. діаграма взаємодії.

2. Вікно, в якому фіксуються усі дії, що виконуються над діаграмами:

A. Standard;
B. Class Diagram Logical View;
C. Browser;
D. Log;
E. ToolBox.

3. Діаграми сценаріїв використовуються для:

A. опису бізнес-процесів предметної області, що автоматизується;
B. визначення вимог до створюваної системи;
C. аналізу апаратної частини;
D. відображення стану об'єкта.

4. Діаграми топологій використовуються для:

A. аналізу апаратних засобів системи;
B. відображення об'єктів системи, задач, виконуваних у системі;
C. відображення поведінки об'єкта;
D. відображення бізнес-процесів об'єкта.

5. Діаграми стану використовуються для відображення:

A. послідовності дій;
B. ієрархії системи;
C. взаємодії об'єкта;
D. поведінки об'єктів у його станах.

6. Діаграми взаємодії дають змогу:

A. шляхом передачі та прийому повідомлень об'єктами-клієнтів та обробки цих повідомлень об'єктами-серверами моделювати поведінку впливу одного об'єкта на інший;

B. відобразити послідовність передачі повідомлень між об'єктами;
C. відобразити ієрархію передачі повідомлень;
D. показати усі повідомлення, що передаються та приймаються для конкретного об'єкта;
E. з різних точок зору проаналізувати взаємодію об'єктів у створюваній системі.

7. Для видалення елемента з моделі використовують команду:

A. Delete;
B. Cut;
C. Undo;
D. Delete from Model.

8. При створенні нової моделі за замовчуванням використовується:

A. шаблон моделі;
B. діаграма класів;
C. посилання на інформаційний ресурс в мережі Інтернет;
D. існуюча модель.

9. У Rational Rose можна імпортувати файли з такими розширеннями:

A. *.ptl;
B. *.pps;
C. *.cat;
D. *.lpt
E. *.sub;
F. *.mdl.

10. У рядку стану відображається інформація про:

A. властивості виділеного об'єкта;
B. діаграму класів;
C. процес завантаження моделі;
D. про відкриття моделі в режимі read-only;
E. специфікацію об'єкта.

11. Використовуючи команди Hide Selected Elements:

A. користувач приховує елементи діаграми: класи, сценарії поведінки, елементи діаграми стану або активності;

B. користувач видаляє елементи діаграми;
C. приховуються елементи, для яких виділений елемент є клієнтом;
D. приховуються Sub-діаграми.

12. Інструмент Use Case — сценарії поведінки відображає:
A. вимоги до системи;
B. зразки поведінки для окремих об'єктів системи;
C. послідовність зв'язаних транзакцій, представлених об'єктами або системою;
D. виконавців системи та задачі, які вони виконують.

МЕТОДИЧНІ ВКАЗІВКИ ДО ПІДГОТОВКИ, НАПИСАННЯ ТА ЗАХИСТУ РЕФЕРАТУ

Реферат є складовою вивчення дисципліни.

Завдання підготовлені відповідно до курсу: "Інформаційні системи і технології в менеджменті" для бакалаврів.

Мета — допомогти студентам засвоїти теоретичні знання, розвинути і сформувати практичні навички використання сучасних інформаційних технологій у своїй майбутній управлінській діяльності. Оформлення і захист реферату має сприяти активному засвоєнню нового матеріалу, виробленню у студентів уміння комплексного використання суміжних дисциплін при вирішенні практичних питань.

За структурою реферат складається з:
вступу;
назви розділу, назви підрозділу;
висновків;
списку літератури;
додатків (за наявності).

У вступі розкривається мета, наводиться загальна характеристика, визначається номер варіанта. У розділах та підрозділах реферату викладається сутність питання, наводяться приклади з посиланням на літературні джерела. Додатки (за їх наявності) подаються наприкінці реферату.

Загальний обсяг роботи не повинен перевищувати 20–30 сторінок машинописного тексту, надрукованого через 1,5 інтервали, рукописне викладення тексту не повинно перевищувати 18–24 сторінок шкільного зошита.

Виконання та оформлення реферату

Студент повинен виконати реферат, розкривши історичні посилки розглядуваної проблеми, відповісти на всі питання як теоретичного плану, так і описати технологію розв'язання практичної задачі, якщо такі передбачені рефератом.

Відповіді на теоретичні питання потребують ретельної роботи з літературою. Крім виписок і конспектування з літературних джерел, наприклад, із Internet, студент повинен зробити висновки. Робота має виконуватися самостійно. У тексті реферату потрібно посилатися на використану літературу. У висновках загалом до реферату розглядають питання економічної доцільності і практичного застосування сучасних інформаційних технологій та обчислювальної техніки в управлінській сфері.

Реферат слід оформляти на стандартних аркушах паперу, зброшурованих у папку. Усі аркуші мають бути пронумеровані. На титульній сторінці потрібно вказати назву вищого навчального закладу, факультет, спеціальність, дисципліну, курс, групу, а також прізвище, ініціали та номер залікової книжки.

На першій сторінці повинні бути вказані варіант контрольної роботи та зміст, який містить питання варіанта, і проставлені номери сторінок, на яких викладено матеріал. На останній сторінці студент підписує роботу і ставить дату. Наприкінці роботи обов'язково треба навести використану літературу. Зшита папка вкладується в поліетиленовий файл. Вона повинна містити дискету з повним текстом, графікою і т. ін. набраного варіанта реферату.

Вибір варіанта реферату

Номер варіанта реферату визначає викладач. Студентам забороняється самостійно змінювати номер варіанта реферату. У разі самостійної зміни варіанта, робота визнається недійсною і підлягає переробці.

Робота, оформлена без виконання наведених вимог, повертається без перевірки на дооформлення.

Індивідуально-консультаційна робота

Індивідуально-консультаційна робота з дисципліни здійснюється у формі консультацій за графіком (одна консультація на два тижні). На консультаціях студентам надаються пояснення з виконання самостійної роботи, підготовки до практичних занять, перевірки та захисту завдань, винесених на поточний контроль тощо.

СПИСОК ЛІТЕРАТУРИ

Основна

1. *Джонс Д. К.* Методы проектирования. — М.: Мир, 1986. — 326 с.
2. *Зиндер Е. З.* Бизнес-реинжиниринг и технологии системного проектирования: Учеб. пособие. — М.: Центр информ. технологий, 1996.
3. *Марка Д. А., МакГоуэн К.* Методология структурного анализа и проектирования. — М.: МетаТехнология, 1993.
4. *Липаев В. В.* Системное проектирование сложных программных средств для информационных систем. — М.: СИНТЕГ, 1999. — 224 с.
5. *Норенков И. П.* Основы автоматизированного проектирования. — М.: Изд-во МГТУ им. Н. Э. Баумана, 2002. — 336 с.
6. *Barker R.* CASE*Method. Entity-Relationship Modelling. Copyright Oracle Corporation UK Limited, Addison-Wesley Publishing Co., 1990.
7. *Barker R.* CASE*Method. Function and Process Modelling. Copyright Oracle Corporation UK Limited, Addison-Wesley Publishing Co., 1990.
8. *Верников Г.* Основные методологии обследования организаций. Стандарт IDEF0 Web-страница: http://www.cfin.ru/vernikov/idef/idef0.shtml
9. *CALS* / ИЛП технологии. Интегрированная Логистическая Поддержка изделий на этапе эксплуатации, Web-страница: http://trim.ru/ru/solut_cspc.html
10. *Калянов Г. Н.* CASE. Структурный системний анализ. — М.: ЛОРИ, 1996. — 242 с.
11. *Вендеров А. М.* CASE-технологии. Современные методы и средства проектирования информационных систем. — М., 1997. — 346 с.
12. *Агафонов В. Н.* CASE-системы и методы спецификации программ // Программные продукты и системы. — 1993. — № 1. — С. 54–57.
13. *Новоженко Ю. В.* Объективно-ориентированые технологии разработки сложных программных систем. — М., 1996.
14. *Трофимов С. А.* CASE-технологии: практическая работа в Rational Rose. — 2-е изд. — М.: Бином-Пресс, 2002. — 288 с.
15. *Вернадов А. М.* Програмирование программного обеспечения экономических информационных систем. — М.: Финансы и статистика, 2002. — 349 с.
16. *Кратчен Филипп.* Введение в Rational Unified Process. — М.: ИД “Вильямс”, 2002. — 240 с.
17. *Дьяконов В. П.* Matlab 6.5 SP1/7 + Simulink 5/6. Основы применения. — М.: СОЛОН-Пресс, 2005. — 800 с.
18. *Засоби* Rational Rose.
19. *Полещук Н.* AutoCad 2002 в подлиннике. Наиболее полное руководство + дискета. — М.: Библион, 2003. — 1200 с.
20. *Ткачев Д.* Самоучитель AutoCad 2002. — М.: Библион, 2003. — 3416 с.
21. *Джеймс Р.* UML. Специальный справочник. — СПб.: Питер, 2002.

22. *Фаулер М., Скотт К.* UML в кратком изложении. — М.: Мир, 1999. — 191 с.
23. *Фаулер М., Скотт К.* UML. Основы. — 2-е изд. — М., 2001.
24. *Ларман К.* Применение UML и шаблонов проектирования. — М.: ИД "Вильямс", 2001. — 498 с.

Додаткова

25. *Oraclei* DBA. — Ч. I: Основы администрирования. Руководство слушателя. — 2002. — Т. 1.
26. *Стив Бобровски.* Oracle8: Архитектура. — М.: ЛОРИ, 1998 с.
27. *Ямпольский Л. С., Калин О. М., Ткач М. М.* Автоматизированные системы технологической подготовки робототехнического производства. — К.: Высш. шк., 1987. — 272 с.
28. *РД-50-34.698-90.* Методические указания. Информационная технология. Комплекс стандартов и руководящих документов на автоматизированные системы. Автоматизированные системы. Требования к содержанию документов. — М., 1992.
29. *ГОСТ* 34.601-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии разработки. — М., 1992.
30. *Ануфриев И., Смирнов А., Смирнова Е.* MATLAB 7.0 в подлиннике. Новая техн. кн., 2005.
31. *Буч Г.* Объектно-ориентированное проектирование с примерами применения. — К.: Диалектика, 1992.
32. *Гайдаржи В. І., Дацюк О. А.* Основи проектування та використання баз даних. — К.: ІВЦ "Вид-во «Політехніка»"; ТОВ "Фірма «Періодика»", 2004. — 256 с.
33. *Chen P. P.* The Entity-Relationship Model: Toward a Unified View of Data ASM Trans. On Database Syst. 1976. — V. 1. — № 1. — P. 9–36.
34. *Буч Г., Румбо Дж., Джекобсон А.* Язык UML. Руководство пользователя. — М.: ДМК, 2000.
35. *Ґенник Дж.* Oracle SQI*PLUS. Карманный справочник. — СПб.: Питер, 2004.
36. *Йордан Э., Аргила К.* Структурные модели в объектно-ориентированном анализе и проектировании. — М.: ЛОРИ, 1999.
37. *Коберн А.* Современные методы описания функциональных требований к системам. — М.: ЛОРИ, 2002.
38. *Луни К., Терью М.* Oracle 9i. Настольная книга администратора. — М.: ЛОРИ, 2002.
39. *Маклаков С. В.* Моделирование бизнес-процесов с AllFusion Process Modeler (BPwin 4.1). — М.: Диалог — МИФИ, 2003.
40. *Мюллер Р. Дж.* Базы данных и UML. Проектирование. — М.: ЛОРИ, 2002.

41. *Смирнов С. Н., Задворьев И. С.* Работаем с Oracle: Учеб. пособие. — М.: Гелиос АРВ, 2002.
42. *Маклаков С. В.* Создание информационных систем с AllFusion Modeling Suite. — М.: Диалог — МИФИ, 2003.
43. *Маклаков С. В.* BPwin та Erwin. CASE-средства разработки информационных систем. — М.: Диалог — МИФИ, 1999.
44. *Маклаков С. В.* Моделирвание бизнес-процессов с BPwin 4.0. — М.: Диалог — МИФИ, 2002.
45. *Методология* функционального моделирования IDEF0. Руководящий документ РД IDEF0-2000. — М.: Госстандарт России, 2000.
46. *Державний* стандарт України. Основні напрямки оцінювання та відбору CASE-інструментів. ДСТУ 3919-1999. — 2000.
47. *Туманов В. Е.* Архитектуры баз данных для крупных промышленных предприятий. — М.: Машиностроитель, 2005. — № 2. — С. 26–32.
48. *Энсор Д., Стивенсон Й.* Oracle. Проектирование баз данных. — К.: BHV, 1999.
49. *Черемных В. Н., Семенов И. О., Ручкин В. С.* Структурный анализ систем: IDEF-технологии. — М.: Финансы и статистика, 2001.
50. *Вайдьянатха Г. К., Дешпанде К., Костелак Д.* Oracle 10.1. Настройка производительности. — М.: ЛОРИ, 2003.
51. *Панащук С. А.* Разработка информационных систем с использованием CASE-системы Silverrun. “СУБД”. — № 3. — 1995.
52. *Горчинская О. Ю.* Designer / 2000 — новое поколение CASE-продуктов фирмы ORACLE. “СУБД”. — № 3. — 1995.
53. *Горин С. В., Тандоев А. Ю.* Применение CASE-средства Erwin 2.0 для информационного моделирования в системах обработки данных. “СУБД”. — № 3. — 1995.
54. *Горин С. В., Тандоев А. Ю.* CASE-средство S —Designer 4.2 для разработки структуры базы данных. “СУБД”. — № 1. — 1996.
55. *DATARUN* Concepts. Computer Systems Advisers Research Ltd., 1994.

Інтернет-ресурси

www.rational.com
www.diasoft.ru
www.idef.com
www.oracle.ru
www.omg.urg
uml.shl.com

ЗМІСТ

Відповідальний за випуск *А. Д. Вегеренко*
Редактор *Т. М. Колина*
Комп’ютерне верстання *О. Л. Тищенко*

Зам. № ВКЦ-3716

Підп. до друку 21.11.08. Формат 60×84/16. Папір офсетний
Друк ротаційний трафаретний.
Ум. друк арк. 2,15. Обл.-вид. арк. 1,73. Наклад 30 пр.

Міжрегіональна Академія управління персоналом (МАУП)
03039 Київ-39, вул. Фрометівська, 2, МАУП
ДП «Видавничий дім «Персонал»
03039 Київ-39, просп. Червонозоряний, 119, літ. ХХ

Свідоцтво про внесення до Державного реєстру
суб’єктів видавничої справи ДК № 3262 від 26.08.2008