МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»
АВТОМОБІЛЬНО-ДОРОЖНІЙ ІНСТИТУТ

Кафедра «Інформаційні системи в економіці»

МЕТОДИЧНІ ВКАЗІВКИ З ДИСЦИПЛІНИ «ОБ'ЄКТНИЙ АНАЛІЗ І МОДЕЛЮВАННЯ СИСТЕМ» (ДЛЯ СТУДЕНТІВ НАПРЯМУ 6.030502 «ЕКОНОМІЧНА КІБЕРНЕТИКА»)

07/ – 2012-11

Горлівка – 2012

МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»
АВТОМОБІЛЬНО-ДОРОЖНІЙ ІНСТИТУТ

«ЗАВЕРДЖУЮ»
Директор АДІ ДВНЗ «ДонНТУ»
М.М. Чальцев

Кафедра «Інформаційні системи в економіці»

МЕТОДИЧНІ ВКАЗІВКИ З ДИСЦИПЛІНИ «ОБ'ЄКТНИЙ АНАЛІЗ І МОДЕЛЮВАННЯ СИСТЕМ» (ДЛЯ СТУДЕНТІВ НАПРЯМУ 6.030502 «ЕКОНОМІЧНА КІБЕРНЕТИКА»)

07/ – 2012-11

«РЕКОМЕНДОВАНО»
Навчально-методична комісія
факультету
«Економіка та управління»
протокол № 3 від 21.12.2011 р.

«РЕКОМЕНДОВАНО»
Кафедра
«Інформаційні системи в економіці»

протокол № 4 від 25.11.2011 р.

Горлівка – 2012

УДК 681.32 (07)

Методичні вказівки до вивчення дисципліни «Об'єктний аналіз і моделювання систем» (для студентів напряму 6.030502 «Економічна кібернетика») [Електронний ресурс] / укладачі.: В. Л. Ніколаєнко, Д. В. Ніколаєнко – Електрон. дані. – Горлівка: ДВНЗ «ДонНТУ» АДІ, 2012. – 1 електрон. опт. диск (CD-R); 12 см. – Систем. вимоги: Pentium; 32 MB RAM; WINDOWS 98/2000/NT/XP; MS Word 2000. – Назва з титул. екрану.

Наведені вимоги до оформлення звіту про виконані роботи з дисципліни «Об'єктний аналіз і моделювання систем», порядок виконання робіт. Також наведені необхідні теоретичні відомості об'єктного аналізу, моделювання й проектування систем. Основні етапи об'єктного аналізу, моделювання й проектування проілюстровані спеціально підібраними прикладами. Методичні вказівки націлені на практичне освоєння теоретичних положень об'єктного моделювання систем.

Укладачі: Ніколаєнко В. Л., к.т.н., доц.
Ніколаєнко Д. В., к.т.н., доц.

Відповідальний за випуск: Ніколаєнко В. Л., к.т.н., доц,

Рецензент: Вовк Л. П., д.т.н., проф.
каф. «Вища математика»

ЕЛЕКТРОННЕ НАУКОВО-МЕТОДИЧНЕ ВИДАННЯ

Ніколаєнко Володимир Леонідович
Ніколаєнко Денис Володимирович

МЕТОДИЧНІ ВКАЗІВКИ З ДИСЦИПЛІНИ «ОБ'ЄКТНИЙ АНАЛІЗ І МОДЕЛЮВАННЯ СИСТЕМ» (ДЛЯ СТУДЕНТІВ НАПРЯМУ 6.030502 «ЕКОНОМІЧНА КІБЕРНЕТИКА»)

Підписано до випуску 2012 р. Гарнітура Times New.
Умов. друк. арк. Зам. № .

Державний вищий навчальний заклад
«Донецький національний технічний університет»
Автомобільно-дорожній інститут
84646, м. Горлівка, вул. Кірова, 51
E-mail: druknf@rambler.ru

Редакційно-видавничий відділ

Свідоцтво про внесення до Державного реєстру видавців, виготовників і розповсюджувачів видавничої продукції ДК № 2982 від 21.09. 2007 р.

ЗМІСТ

ВСТУП

Сучасною методологією аналізу й проектування складних систем у даний час є об'єктний аналіз і об'єктне проектування, які істотно доповнюють існуючі методології та відкривають можливості широкого використання CASE технологій розробки моделей систем.

Дисципліна складається з таких розділів:
1) графічні примітиви UML стандарту;
2) об'єктне моделювання економічних систем;
3) об'єктне проектування програмних моделей систем.

Мета й завдання методичних вказівок полягає в ознайомленні студентів з теоретичними основами об'єктного аналізу, моделювання і проектування програмних моделей економічних систем, придбанні практичних навичок аналізу й розробки об'єктної моделі економічної системи й проектування її програмної моделі, керованої виявленим у результаті об'єктного аналізу потоком подій.

Основними завданнями методичних вказівок є:
1) вивчення теоретичних основ і принципів об'єктного аналізу й моделювання економічних систем;
2) вивчення теоретичних основ і принципів проектування програмних моделей економічних систем;
3) придбання практичних навичок використання об'єктного моделювання та проектування програмних моделей економічних систем.

У результаті виконання індивідуальних завдань студенти повинні

– знати:

1) теоретичні основи та принципи об'єктного аналізу й моделювання економічних систем;
2) графічні примітиви UML стандарту;
3) основи та принципи об'єктного проектування програмних моделей економічних систем;

– мати навички:

1) використання примітивів UML стандарту при розробці об'єктних моделей економічних систем;

2) використання технології об'єктного проектування програмних моделей економічних систем.

1 ВИМОГИ ЩОДО ОФОРМЛЕННЯ ЗВІТУ

1.1 Структурні елементи звіту

Звіт про виконані роботи складається з текстового файлу звіту й VB.NET проекту системи.

Нижче приведена рубрикація текстового файлу звіту.

Приклад 01 «Аналіз прецедентів».

Приклад 02 «Аналіз класів».

Приклад 03 «Аналіз послідовностей подій».

Приклад 04 «Аналіз потоку подій».

Приклад 05 «Аналіз станів об'єктів».

Приклад 06 «Проекція системи в простір станів».

Приклад 07 «Аналіз кооперації».

Приклад 08 «Аналіз діяльності».

Приклад 09 «Аналіз модулів».

Приклад 10 «Аналіз топології».

Приклад 11 «Визначення специфікацій класів».

Приклад 12 «Тестування потоку подій».

Приклад 13 «Тестування діяльності об'єктів».

Електронна копія звіту подається у файлі

ObjModel Ek_00a Sidorov I.p. Report.doc

Жорстка копія текстового файлу звіту складається з роздрукованих на принтері титульного аркуша та аркуша завдань.

Для захисту роботи студент надає викладачеві проект системи, титульний аркуш та аркуша завдань.

2 ПОРЯДОК ВИКОНАННЯ РОБІТ

Крок 1 «Отримання завдання».

Індивідуальні завдання згідно варіанта слід отримати у викладача.

Крок 2 «Побудова фрагмента дерева каталогів».

Всі файли слід розміщувати в каталозі `ObModel`. Нижче наведений фрагмент дерева каталогів.

```
E:
|__Student
  |__Ек_00a
     |__Sidorov I.P.
        |__ObAnPrSys
        |__
```

Крок 03 «Підготовка шаблону файлу звіту».

ObAnPrSys Ek_00_a Sidorov P.N. Report.doc

Крок 04 «Аналіз прецедентів».

Крок 05 «Аналіз класів».

Крок 06 «Аналіз послідовностей подій».

Крок 07 «Аналіз потоку подій».

Крок 08 «Аналіз станів об'єктів».

Крок 09 «Проекція системи в простір станів».

Крок 10 «Аналіз кооперації».

Крок 11 «Аналіз діяльності».

Крок 12 «Аналіз модулів».

Крок 13 «Аналіз топології».

Крок 14 «Визначення специфікацій класів».

Крок 15 «Тестування потоку подій».

Крок 16 «Тестування діяльності об'єктів».

3 ВКАЗІВКИ ДО ВИКОНАННЯ РОБІТ

3.1 Парадигми UML стандарту

А Словник мови.

Б Правила над словником.

В Механізми.

Словник мови
- |_ Суть
- |_ Стосунки
- |_ Діаграми
- |_ Анотуюча суть

Суть
- |_ Структурна суть
- |_ Поведінкова суть
- |_ Групуюча суть

Структурна суть
- |_ Прецеденти
- |_ Класи
  - |_актори
  - |_сигнали
  - |_утіліти
- |_ Інтерфейси
- |_ Активні класи
  - |_процеси
  - |_ніті
- |_ Кооперації
- |_ Компоненти
- |_ Вузли

Анотуюча суть
- |_ Примітки

Поведінкова суть
- |_ Взаємодія
- |_ Автомати

Групуюча суть
- |_ Пакети

Стосунки
- |_ Залежність
- |_ Асоціація
- |_ Узагальнення
- |_ Композиція

Діаграми

|_ Варіантів використання (Прецедентів) User case
|_ Класів Class
|_ Кооперації Collaboration
|_ Послідовності Sequence
|_ Станів Statechart
|_ Діяльності Activity
|_ Компонентів Component
|_ Розгортання Deployment

Правила над словником

Правила над словником – це правила використання графічних примітивів в діаграмах.

Механізми

Механізми – це механізми побудови діаграм.

3.2 Графічні примітиви UML стандарту

Прецедент

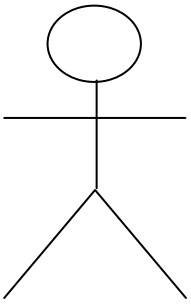
Прецедент – це послідовність дій, значущих для деякого актора.

Позначення:

Актор

Актор – це зовнішня по відношенню до системи суть.

Позначення:



Види «зовнішніх акторів»:

1) Користувач. 2) Інша система.
3) Час.

Клас

Клас являє собою статичну частину моделі, відповідну концептуальному або фізичному елементу системи.

Позначення

Активний клас

Активний клас – це клас, що відрізняється тим, що його об'єкти активні під час роботи інших об'єктів системи й володіють потоком або процесом з можливістю генерації дій, що управляють.

Інтерфейс

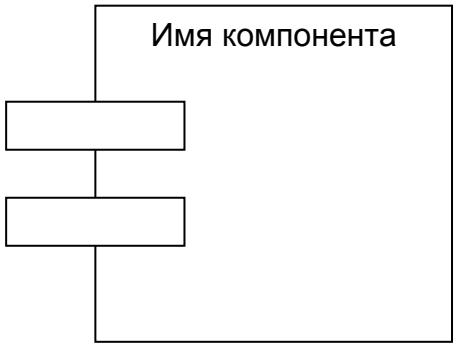
Інтерфейс – це сукупність обов'язків класу, його сервіс, набір послуг, які він надає.

Кооперація

Кооперація – це сукупність ролей, які грає клас в своїх взаємодіях так, що кооперативний ефект не є просто сумою ролей.

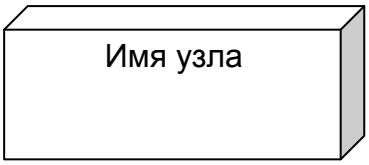
Компонент

Компонент – це фізичний образ логічних елементів і їх інтерфейсів.



Вузол

Вузол – це фізичний образ комутуючого ресурсу, що володіє власною пам'яттю й засобами обробки.

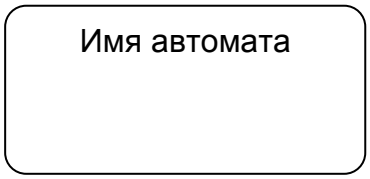


Взаємодія

Взаємодія – це поведінка, в основі якої лежить повідомлення (подія).

Автомат

Автомат – це алгоритм поведінки, що відображає послідовність станів, через який об'єкт або взаємодію проходять впродовж свого життєвого циклу у відповідь на потік подій.

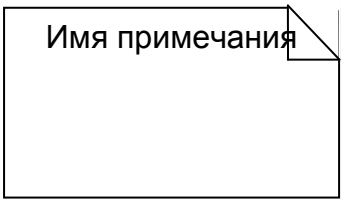


Пакет

Пакет – це організаційна, групуюча суть концептуального рівня проекту, що включає структурну, поведінкову суть або інші пакети.

Примітка

Примітка – це пояснення до елементу проекту.



Відношення

- |_ Залежність
- |_ Асоціація
- |_ Узагальнення
- |_ Агрегація
- |_ Композиція

Залежність

Залежність – це таке відношення між суттю, коли зміна однією може відбитися на іншій.

Ім’я залежності

----------->

Початкова суть Залежна суть

Асоціація

Асоціація – це зв'язок рівноправної суті.

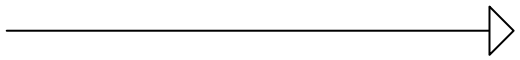
Ім'я асоціації або Ім'я асоціації

_____ _____

1-а суть 2-а суть 1-а суть 2-а суть

Узагальнення

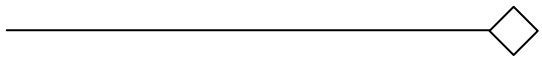
Узагальнення – це спадкоємство однієї суті іншою.



Нащадок суть Батько суть

Агрегація

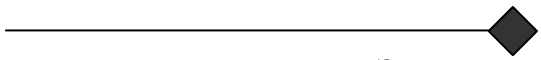
Агрегація – це коли одна суть є частиною іншої суті.



Частина суть Загальне суть

Композиція

Композиція - це агрегація, коли частина не може існувати у відриві від цілого.



Частина суть Загальне суть

3.3 Діаграми UML стандарту

Діаграми використовуються для графічного відображення різних аспектів системи з метою полегшення сприйняття й розуміння роботи системи та відображають структуру, стани об'єктів системи та її архітектуру. Враховуючи стандартизований набір графічних примітивів допускається використання CASE технологій.

Види діаграм:

1) Варіантів використання (прецедентів)	User case
2) Класів	Class
3) Кооперації	Collaboration
4) Послідовності	Sequence
5) Станів	Statechart
6) Діяльності	Activity
7) Компонентів	Component
8) Розгортання	Deployment

3.4 Приклад 01 «Аналіз прецедентів»

Діаграми прецедентів («Варіантів використання» User Case)

Варіант використання - це сценарій переробки даних, керований потоком подій.

Варіант використання («Прецедент») – це послідовність дій, що виконуються у відповідь на подію, ініційовану користувачем, іншою системою, часом і т.п. – «зовнішнім актором».

Приклад

Система – «Приватний підприємець».

Варіант використання – «Здача виручки в банк».

Що дають діаграми «Варіантів використання»?

Відповідь

Перелік сервісів системи, тобто набір її функціональностей.

Схема документування «Варіанту використання».

Варіант використання «*Назва»*.

Короткий словесний виклад сенсу варіанту прецеденту.

Передумови

Надається короткий перелік того, що треба робити й мати для роботи перед початком варіанту використання.

Основний сценарій

Надається логістична модель сценарію прецеденту.

Альтернативний сценарій 1

Надається логістична модель альтернативного сценарію прецеденту.

Постумова

Надається короткий перелік того, чим треба робити і мати для завершення прецеденту.

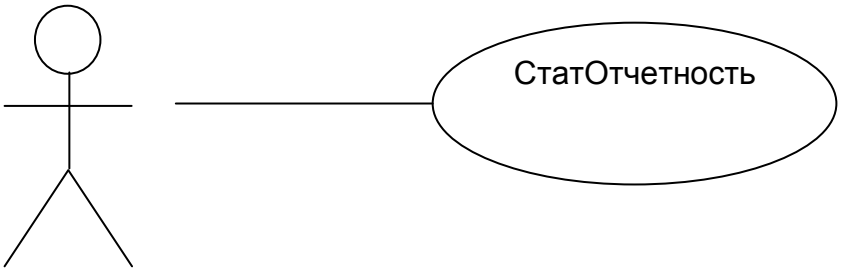
Відношення на діаграмах прецедентів

Асоціація

Асоціація використовується для позначення зв'язку між актором і прецедентом.

Позначення:

Приклад:

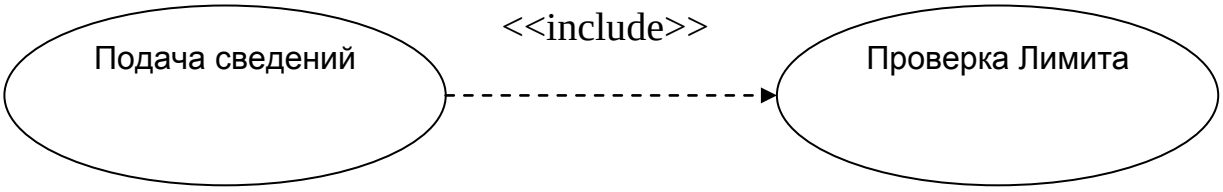


Включення

Включення використовується для вказівки зв'язку між базовим прецедентом і тим прецедентом, сценарій якого є частиною сценарію базового прецеденту. Використовується слово <<include>>.

Позначення:

Приклад

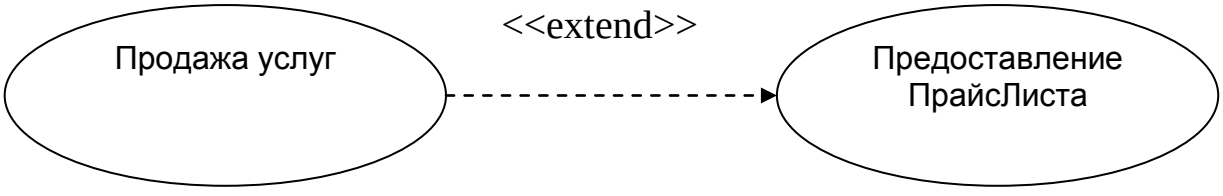


Розширення

Розширення використовується для вказівки зв'язку між прецедентом і тим прецедентом, сценарій якого може бути використаний базовим прецедентом при виконанні деяких умов. Використовується слово <<extend>>.

Позначення:

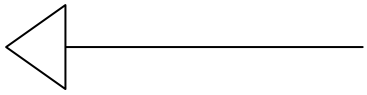
Приклад:



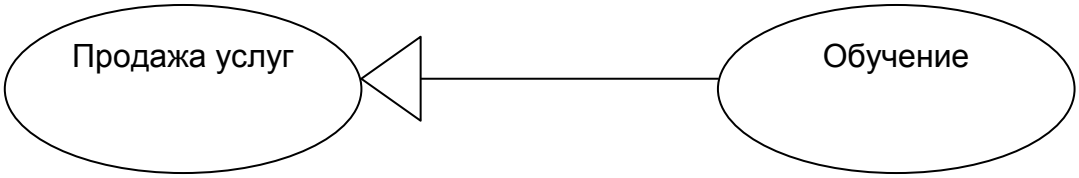
Узагальнення

Узагальнення використовується для вказівки зв'язку між прецедентом-предком і прецедентом-нащадком, сценарій якого успадковує сценарій прецеденту предка з, можливо, його доповненням.

Позначення:



Приклад:



Стрілка вказує на предка.

Як робити діаграми «Прецедентів» («Варіантів використання»?).

Відповідь:
Розробник, представивши себе підприємцем («ЧаП»):

ПЕРШЕ – міркує:
«Треба виконати процедуру «Звіт у податковій службі».

«Треба виконати процедуру «Подача відомостей в «ПенсФонд».

«Треба виконати процедуру «Подача відомостей в «ЦенрЗанят».

«Треба виконати процедуру «Подача відомостей в

«СоцСтрахНесчСлучай».

«Треба виконати процедуру «Подача відомостей в

«СоцСтархПотериТрудоспособности».

«Треба виконати процедуру «Подача відомостей в «ГорСтатистику».

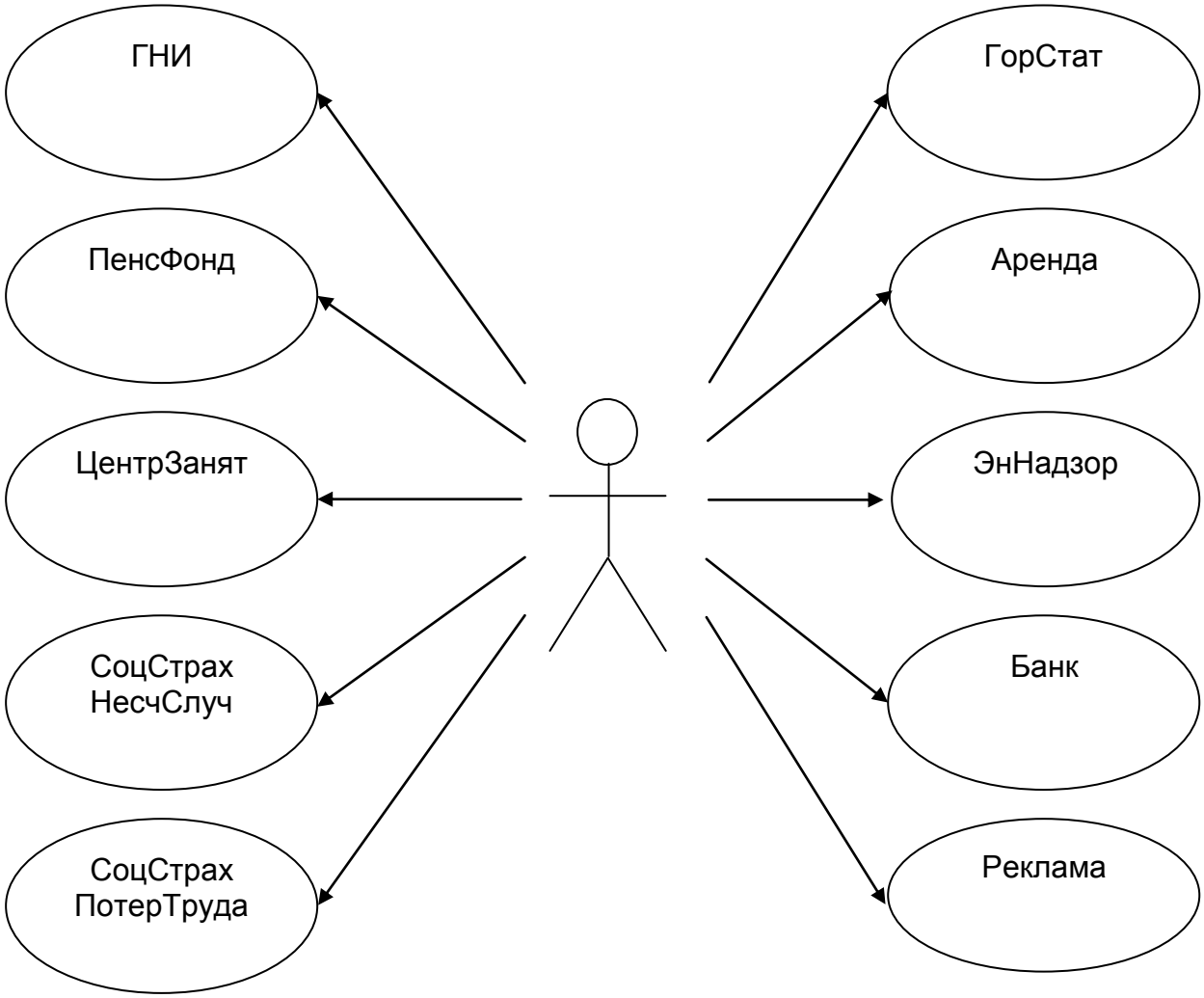
«Треба виконати процедуру «Аренда».

«Треба виконати процедуру «Подача відомостей в «ЕнергоНагляд».

«Треба виконати процедуру «Здача виручки в банк».

«Треба виконати процедуру «Отримання зарплати», «Реклама».

ДРУГЕ – малює діаграму «Прецедентів».



3.5 Приклад 02 «Аналіз класів»

Діаграми «Класів» (Class).

Діаграми «Класів» використовуються для відображення набору класів і відносин між ними.

Класи зображають з їх атрибутами – властивостями, методами.

Представляють логічний аспект системи.

На етапі аналізу, діаграми класів використовуються для виявлення ролей і обов'язків суті в поведінці системи.

На етапі проектування, діаграми класів використовуються для відображення структури класів, тобто архітектури системи.

Види класів

«Граничні класи» (Boundary classes)

«Граничні класи» – це класи, що забезпечують взаємодію системи із зовнішнім світом: форми, звіти, інтерфейси з HardWare, інтерфейси з іншими системами.

«Класи суті» (Entity classes)

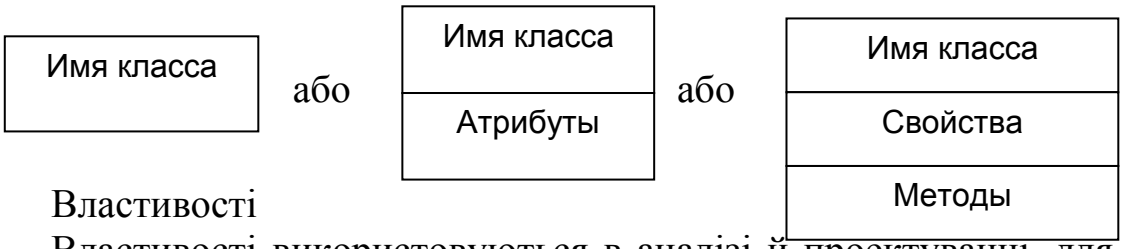
«Класи суті» – це класи, що виражають собою основні абстракції системи. Зазвичай для них створюють бази даних.

«Класи, що управляють» (Control classes)

«Класи, що управляють» – це класи, контролюючі послідовність дій сценарію. Такі класи можуть управляти декількома варіантами використання. Це класи:менеджерів безпеки, ресурсів, процесів, подій, помилок і так далі.

Позначення в діаграмах класів:

Клас



Властивості

Властивості використовуються в аналізі й проектуванні, для виразу властивостей класів.

Позначення

або A – тільки ім'я
або :C – тільки клас
або A:c – ім'я класа
або A:c = e – ім'я, клас і значення за умовчанням.

Методи

Методи використовуються в аналізі й проектуванні для виразу послуг, що надаються класом.

Позначення:

або N() – тільки ім'я
або R N(аргументи) – тільки клас

R – клас значення, що повертається (тип, що повертається)
N – ім'я операції.

Аргументи – це аргументи в класичному уявленні.

Атрибут – це властивість, метод.

Видимість атрибуту (Attribute visibility):

Public
Public – доступний всім класам.

Private
Private – доступний тільки усередині класам.

Protected
Protected – доступний усередині класам і всім нащадкам.

Види методів:

1. Методи управління (Manager operations)
Методи управління використовуються для створення і знищення об'єктів.

2. Методи доступу (Access operations)
Методи доступу використовуються для доступу до атрибутів класу.

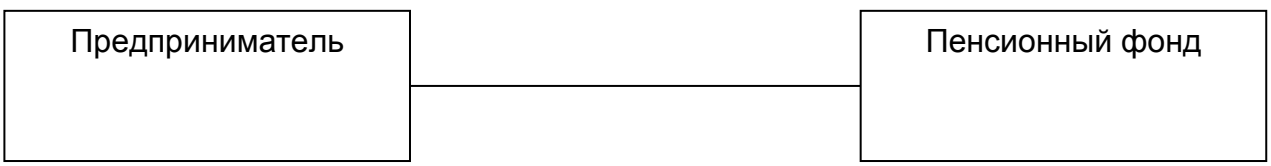
3. Допоміжні методи (Helper operations)
Допоміжні методи використовуються для виконання закритої функціональності класу.

4) Методи реалізації (Implementor operations)
Методи реалізації використовуються для обробки подій.

Типи зв'язків між класами

Асоціація
Асоціація використовується для вказівки зв'язку між класами, що асоційовані деякою ідеєю.
Позначення:

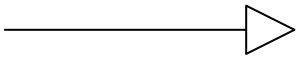
Приклад:



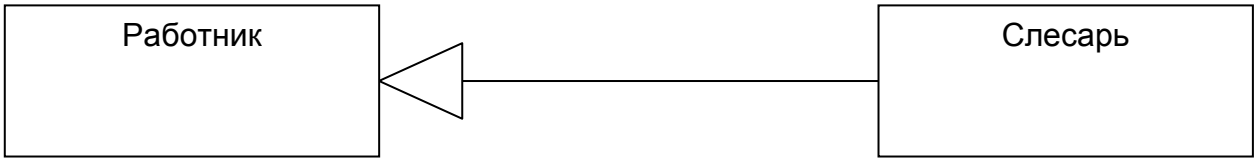
Узагальнення (спадкоємство «is-a»).

Узагальнення використовується для вказівки зв'язку між нащадком класом і предком класом, атрибути якого успадковує клас нащадок з, можливо, їх доповненням.

Позначення:



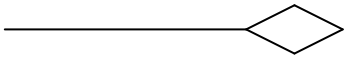
Приклад:



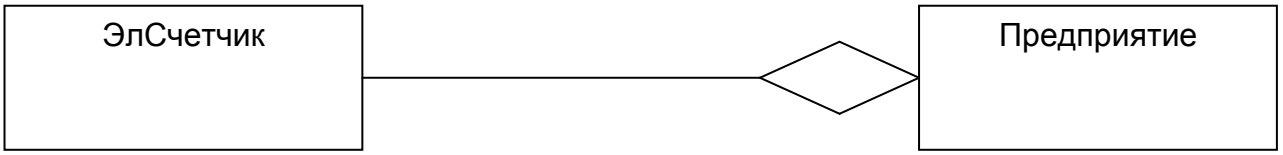
Зауваження
Стрілка указує на предка.

Агрегація (Ціле – частина «part-off»)
Агрегація використовується для вказівки зв'язку між класами, коли один клас реалізує частину функцій іншого класу.

Позначення:



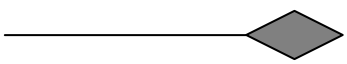
Приклад:



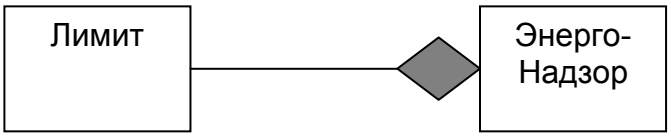
Зауваження
Стрілка указує на клас-контейнер.

Композиція
Композиція використовується для вказівки зв'язку між класами, коли один клас реалізує частина функцій іншого класу, причому клас-частина не може існувати окремо від класу-контейнера.

Позначення:



Приклад:

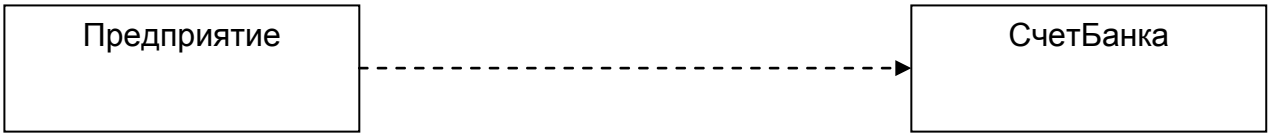


Зауваження
Стрілка указує на клас-контейнер.

Залежність
Залежність використовується для вказівки зв'язку між класами, коли зміна стану одного класу може спричинити зміну стану іншого класу.

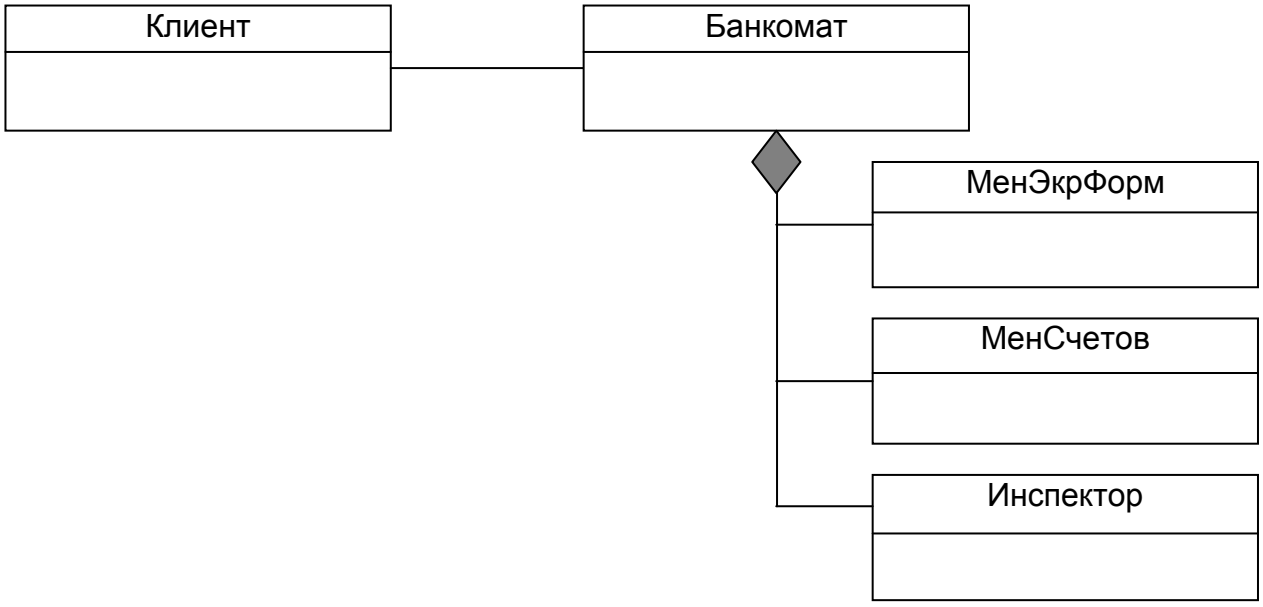
Позначення:

Приклад:



Зауваження
Стрілка указує на залежний клас.

Приклад діаграми класів
Діаграма класів системи «Банкомат»



3.6 Приклад 03 «Аналіз сценаріїв і послідовностей подій»

Діаграми «Взаємодій» використовуються для відображення виконання сценарію варіанту використання (процедури).

На «Діаграмах взаємодій» зображають об'єкти й повідомлення, якими вони обмінюються один з одним.

Види повідомлень:

Повідомлення – це подія, виклик методу, читання значення властивості, установка значення властивості.

1 Повідомлення (Message).

Повідомлення – це засіб, яким «ОбъектИсточник» запрошує у «ОбъектаПолучателя» виконання його операції.

2 Інформаційне повідомлення (Informative)

Інформаційне повідомлення – це засіб, яким «ОбъектаПолучатель» забезпечується інформацією для зміни свого стану.

3 Повідомлення-запит (Interrogative).

Повідомлення-запит – це засіб, яким запрошується інформація про «ОбъектаПолучателя».

4 Імперативне повідомлення (Imperative).

Імперативне повідомлення – це засіб, яким запрошується виконання дії «ОбъектаПолучателя».

Два види діаграм взаємодії:

а) діаграми послідовності (Sequence diagrams);

Діаграми послідовності відображають швидше часовий потік подій сценарію варіанту використання.

б) корпоративні діаграми (Collaboration diagrams);

Корпоративні діаграми відображають швидше міжоб'єктний потік подій сценарію варіанту використання.

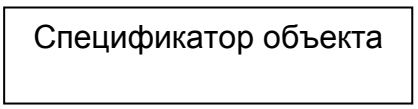
Діаграми послідовності

На ранніх ітераціях аналізу концентруються на подіях, оскільки вони краще показують межі сценаріїв.

Діаграми послідовності (Sequence diagrams) використовуються для відображення взаємодії між об'єктами на основі повідомлень в контексті тимчасових особливостей потоку повідомлень. Відображають часовий потік подій сценарію варіанту використання.

Позначення:

Об'єкт



Лінія життя

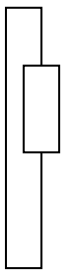
Лінія життя використовується для позначення періоду часу, протягом якого об'єкт існує в системі.

Фокус управління

Фокус управління – це об'єкт системи, що володіє управлінням, тобто що виконує деякі дії, має фокус управління (focus of control).

Вкладений фокус управління

Вкладений фокус управління – це об'єкт, що посилає повідомлення собі (повідомлення рефлексії) і що виконує відповідну дію, має рекурсивний фокус або вкладений фокус.



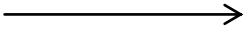
Повідомлення

Повідомлення використовується для специфікації зв'язку між об'єктами.

Синхронне повідомлення

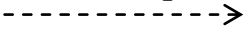
Синхронне повідомлення не тільки передає одержувачеві інформацію, але, можливо, і припускає деяку дію одержувача. Об'єкт-джерело чекає закінчення операції об'єкта-одержувача.

Асинхронне повідомлення



Асинхронне повідомлення передається в довільні моменти часу. Зазвичай, не активізує фокус управління одержувача. Об'єкт-джерело не чекає закінчення операції об'єкта-одержувача.

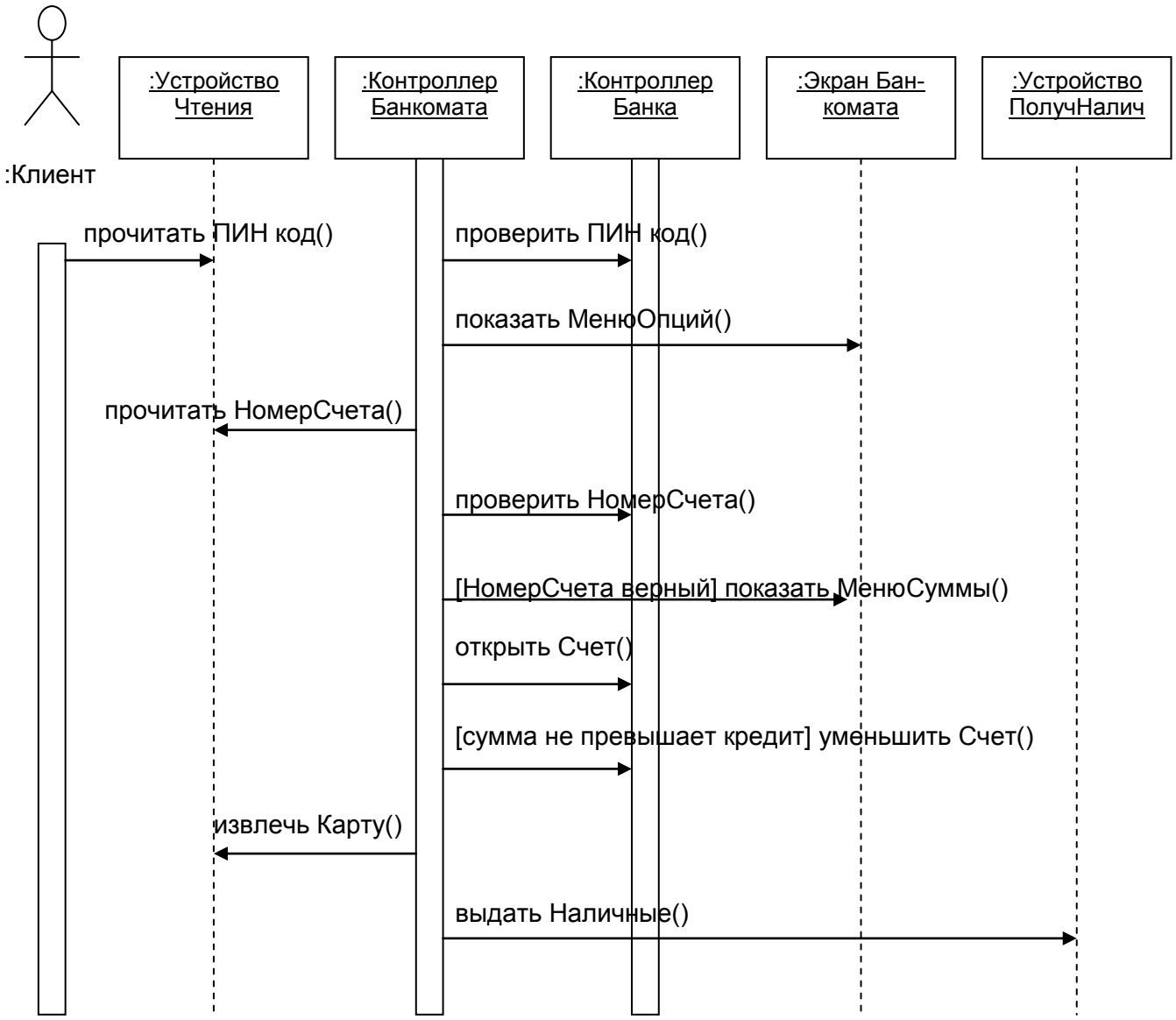
Поворотне повідомлення



Поворотне повідомлення – це об'єкт-джерело отримує повідомлення про закінчення операції об'єкта-одержувача.

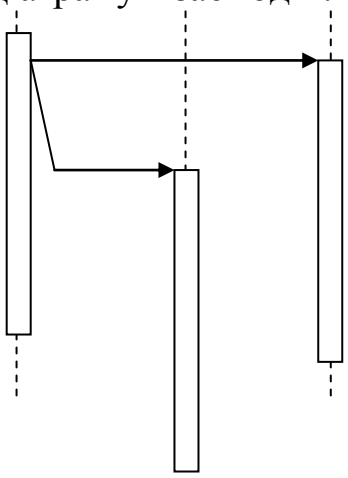
Приклад діаграми послідовності

Система «Банкомат». Сценарій «Основний».



Галуження потоку управління

Галуження потоку управління використовується для відображення галуження Потоку управління у разі, коли кількість галужень не захаращує діаграму Взаємодій.



3.7 Приклад 04 «Аналіз потоку подій»

На додаток аналізу послідовності подій корисно робити аналіз «Потоку подій» (Flow of Events), графічним представленням якого є «Діаграма потоку подій» і «Тимчасова діаграма потоку подій».

Діаграма потоку подій відображає результат аналізу потоку подій сценарію, але на відміну від діаграми послідовності подій UML стандарту, вона не переобтяжена графічними примітивами й наочніше представляє потік подій сценарію.

Діаграма «Потоку подій» (Flow of Events)

Діаграма «Потоку подій» відображають Потік подій, лежачий в основі взаємодії суті системи.

У діаграмі «Потоку подій» використовуються наступні графічні примітиви:

Вісь об'єктів

Позначення:

– | – – – – – – | – – – – – – | – – – – – – | – – – – – – | – – – –→

Об'єкт

Об'єкт специфікує об'єкт системи.

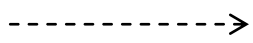
Позначення:

A B C D E

Використовується текстовий стереотип формату: <<Ім’яОб'єкта>>.

Подія
Указує на об'єкт джерело події й на об'єкт одержувач події.

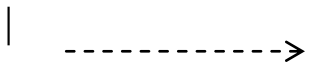
Позначення:



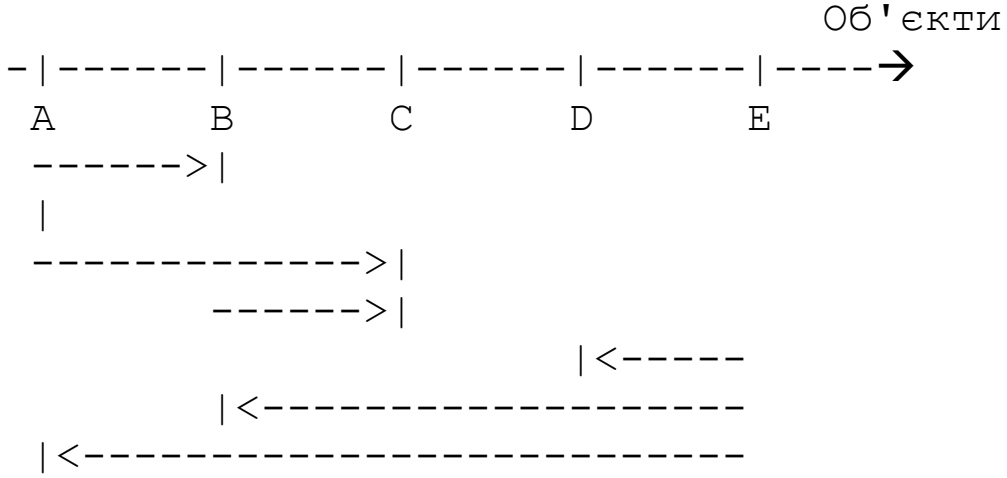
Активність об'єкту

У діаграмі потоку подій тривалість активності об'єкту не істотна, а в тимчасовій діаграмі потоку подій тривалість активності об'єкту відіграє помітну роль і дозволяє обчислити тривалість активності кожного об'єкту при виконанні сценарію.

Позначення:



Приклад « Діаграма потоку подій»:

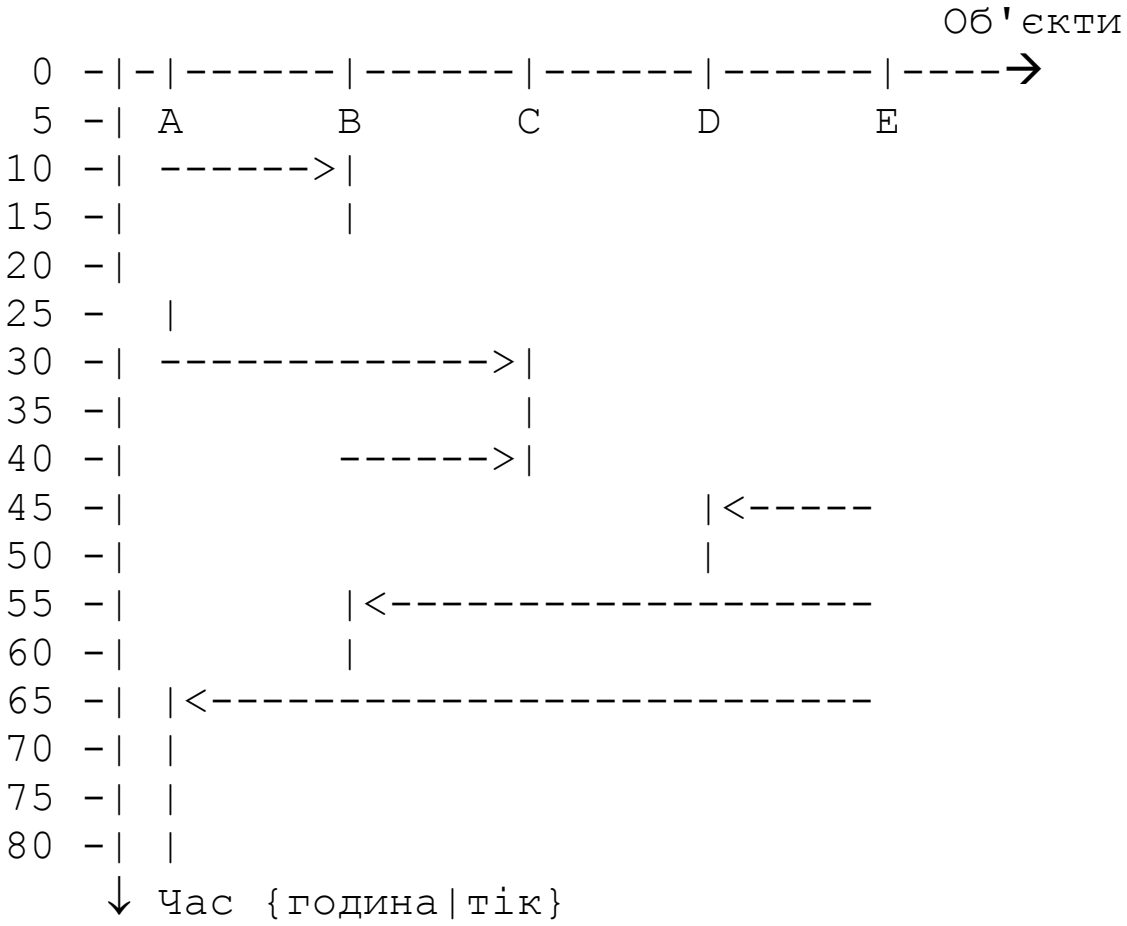


Тимчасова діаграма потоку подій використовується для тимчасового масштабування потоку подій.

Тимчасова діаграма потоку подій відображає результат аналізу потоку подій сценарію, але на відміну від діаграми потоку подій, вона представляє потік подій, масштабований у часі.

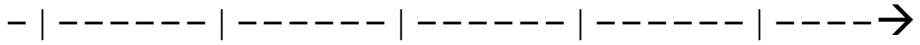
Використовується, зокрема, для обчислення вартісної характеристики сценарію прецеденту на основі наперед обчислених вартостей активності кожного об'єкту.

Приклад «Тимчасова діаграма потоку подій»:

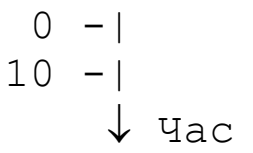


У тимчасовій діаграмі потоку подій використовуються наступні графічні примітиви:

– вісь об'єктів

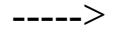


- вісь часу



– імена об'єктів

– подія



– активність об'єкту

3.8 Приклад 05 «Аналіз станів об'єктів»

Діаграма «СтанівОб'єктів» (SateChart).

«СтанівОб'єктів» – це миттєвий знімок стану об'єктів системи.

Діаграми станів створюють, коли об'єкт може знаходитися в різних станах і його поведінка в них є різною.

Використовується для відображення простору станів, в яких може знаходитися об'єкт і процес зміни станів при настанні події.

Використовується при аналізі системи для показу семантики основних і другорядних сценаріїв системи.

Об'єкт-клієнт той, який викликає характеристики іншого об'єкту.

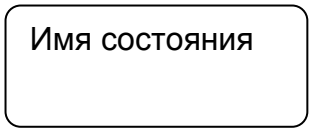
Об'єкт-сервер той, характеристики якого викликаються іншим об'єктом.

Позначення в діаграмах станів і переходів об'єктів

Стан

Стан – це підсумковий результат поведінки системи. Кожен стан може мати ім'я. Безліч станів кінцева.

Позначення:



Ім'я стану пишуть з великої літери. Носить закінчений характер. Рекомендується використовувати дієслово в теперішньому часі – «Дзвонить», «Друкує», «Чекає».

Список дій у стані:

1 Вхідні дії (Entry actions)

Вхідні дії – це дії, які виконуються, коли об'єкт переходить в даний стан.

2 Діяльність (Do activity)

Діяльність – це дії, які виконуються, коли об'єкт знаходиться в даному стані.

3 Вихідних дії (Exit actions)

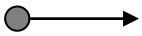
Вихідних дії – це дії, які виконуються, коли об'єкт виходить з даного стану.

4 Включення дій підавтомата (Include)

Включення дій підавтомата – це дії, які виконуються автоматом, включеним в діяльність.

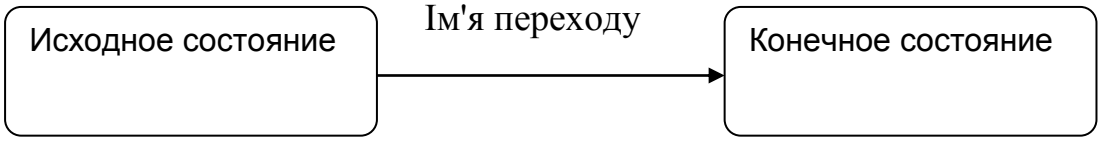
Початковий стан і кінцевий стан

Початковий стан і кінцевий стан використовується для вказівки початка або закінчення процесу зміни станів системи.



Перехід
Перехід відображає факт зміни станів системи.

Позначення:



Ім'я переходу – це рядок тексту формату:
<сигнатура події>[<сторожова умова>}<вираз дії>

Сигнатура події – це рядок тексту формату:
<ім'я події>(<список параметрів через кому >)

Терми формату можуть бути відсутніми.

З переходом асоційовані:

Подія
Подія лежить в основі переходу.

Якщо подія-тригер («да-ні»), то перехід називають тригером. Обов'язково іменується.

Якщо подію позначає завершення дії, то перехід називають нетригером. Не іменується.

Сторожова умова (Обмеження) Guard conditions.

Сторожова умова – це умови логічного типу, що визначають, коли перехід може або не може відбутися.

Дії (Actions)

Дії – це дії, які виконуються під час переходу в новий стан.

Складений стан або суперстан або стан-композит – це стан, який включає інші стани – підстани.

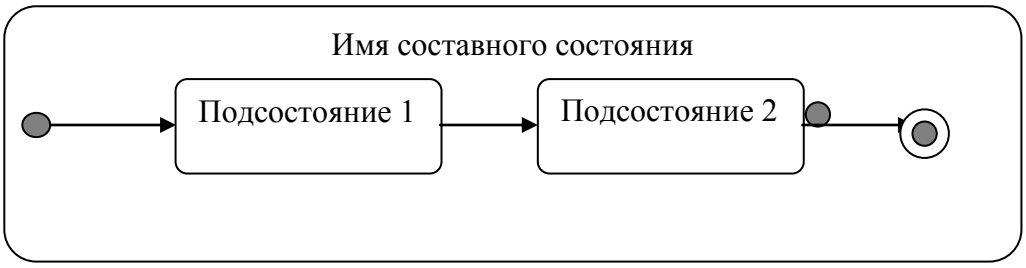
Коли стан (суперстан) містить вкладені стани (підстани), говорять про вкладені стани. Використовується для абстрагування від неістотних станів.

Підстан

Підстан – це стан, який входить в склад суперстани.

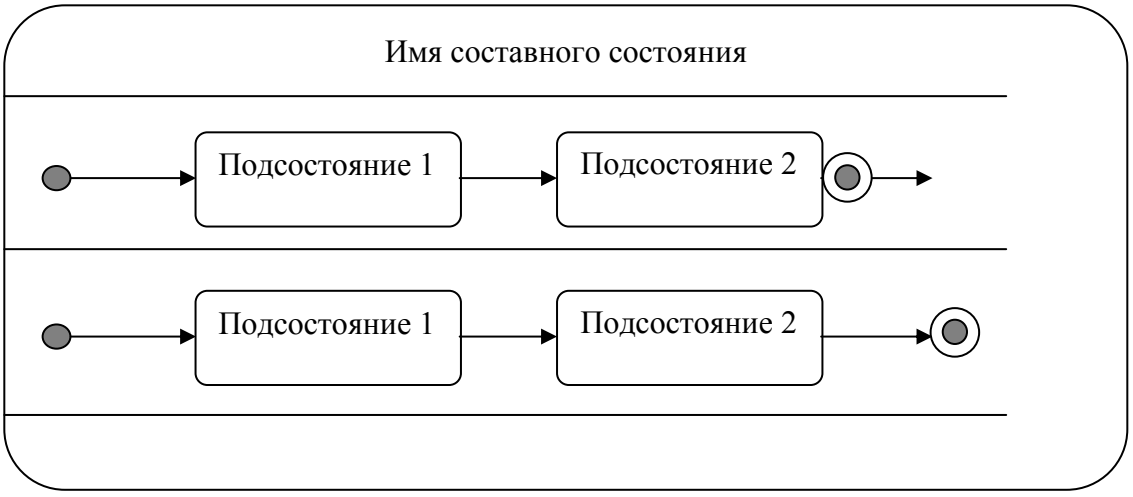
Послідовні підстани

Послідовні підстани – це підстани, які виконуються послідовно.



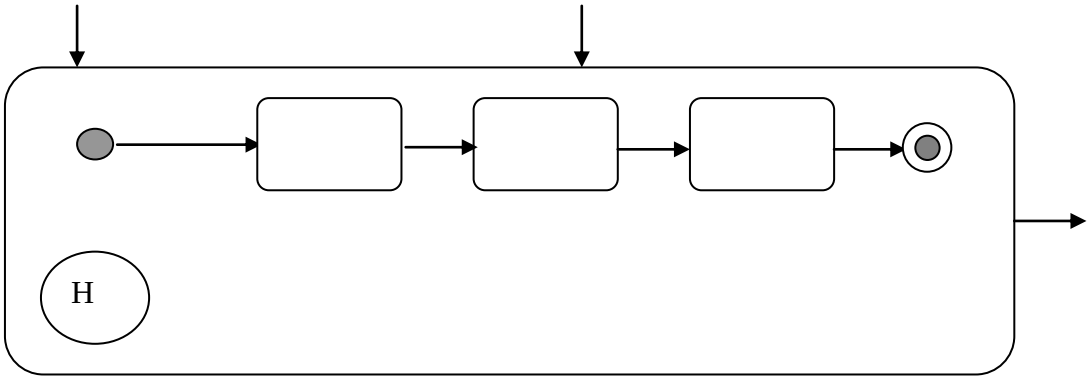
Паралельні підстани

Паралельні підстани – це підстани, які можуть виконуватися паралельно.



Історичний стан

Якщо новий вхід в стан залежить від попереднього стану, говорять про історію станів.



Про історію підстанів говорять, коли вихід з складеного стану відбувається не після його закінчення, а з його середини.

Неглибокий історичний стан

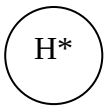
Якщо запам'ятовується вкладений стан, на якому відбувся вихід з складеного стану, і подальше повернення в складений стан відбувається на вкладений стан, що запам'ятав, говорять про неглибокий історичний складений стан.

Позначення:

Глибокий історичний стан

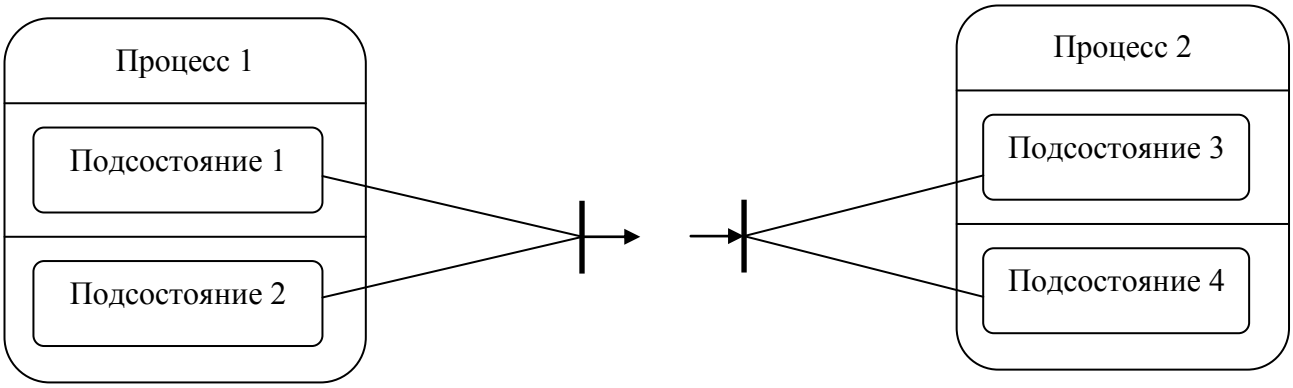
Якщо запам'ятовується вкладений стан, на якому відбувся вихід з складеного стану, і подальше повернення в складений стан відбувається на вкладений стан, що запам'ятав, то говорять про глибокий історичний складений стан у тому випадку, коли вкладений стан є знову складений стан.

Позначення:



Переходи між паралельними станами

Якщо цільовий стан має декілька початкових, або один початковий стан має декілька цільових, то говорять про паралельні переходи між станами й використовують наступні позначення:



Злиття переходів відбувається, якщо є подія-тригер для всіх станів, спрацьовування якої приводить до одного стану.

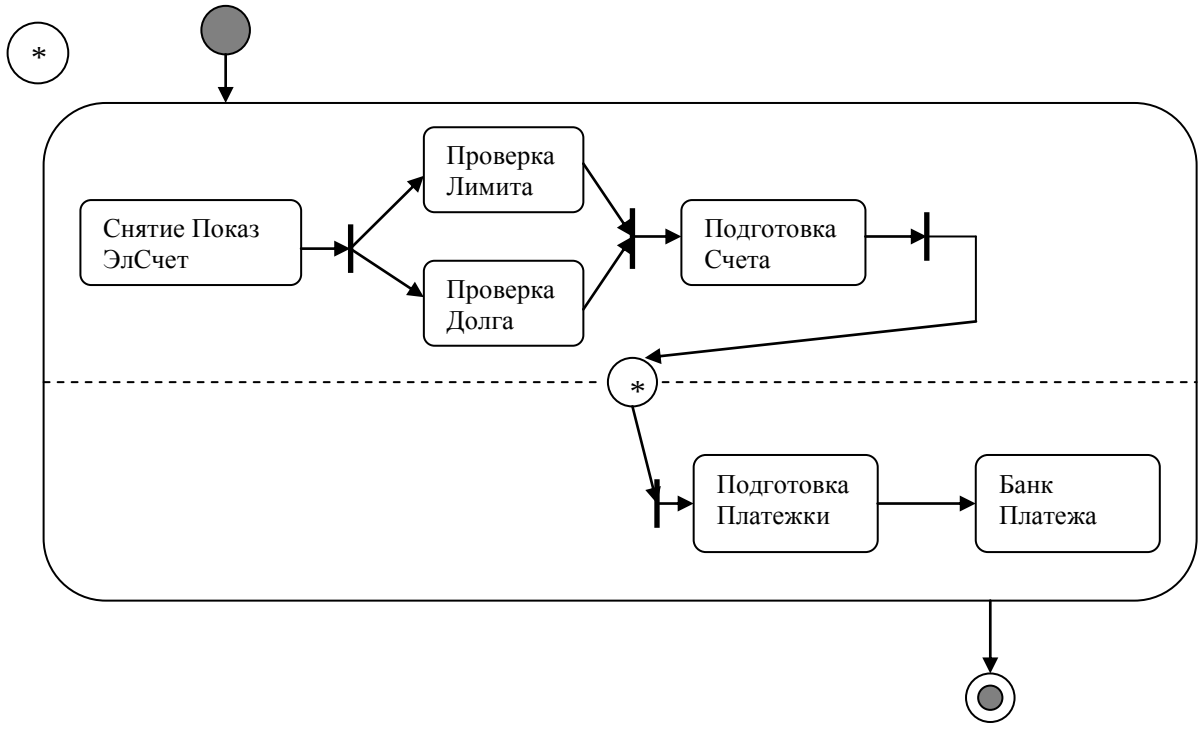
Переходи між складеними станами

Перехід в складений стан може бути на всі або не всі вкладені стани. Вихід із складеного стану може бути з одного або жодного з вкладених станів. При цьому допустимі будь-які переходи між вкладеними станами усередині складеного стану.

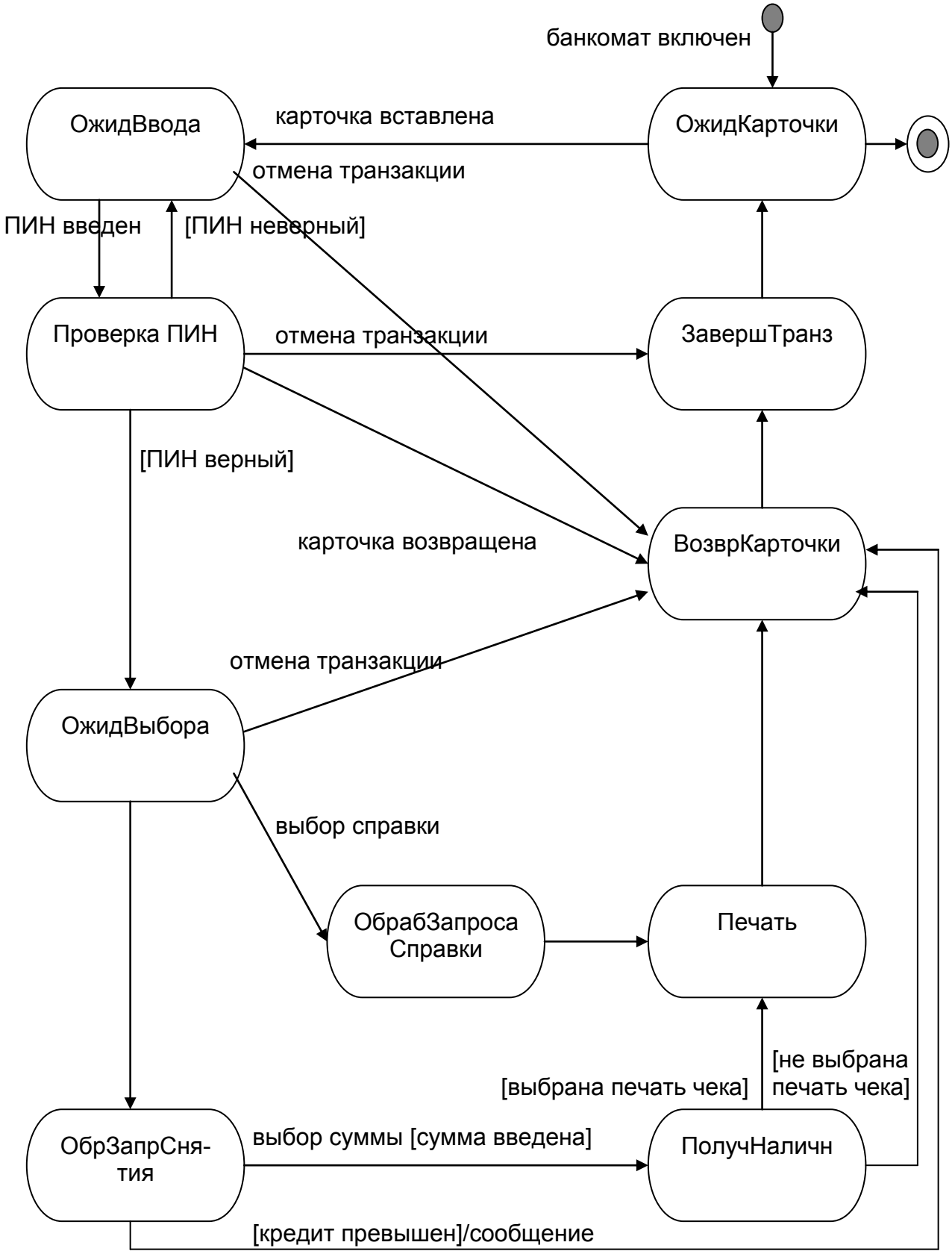
Синхронізуючі стани

При злитті або розділенні переходів використовують, при необхідності, синхронізуючі стани з метою узгодження в часі процесів – «ПодготовкаСчетаОплатыЭлектроенергии» можлива лише після перевірки «ЛимитаПотребленияЭлектроенергии».

Позначення:



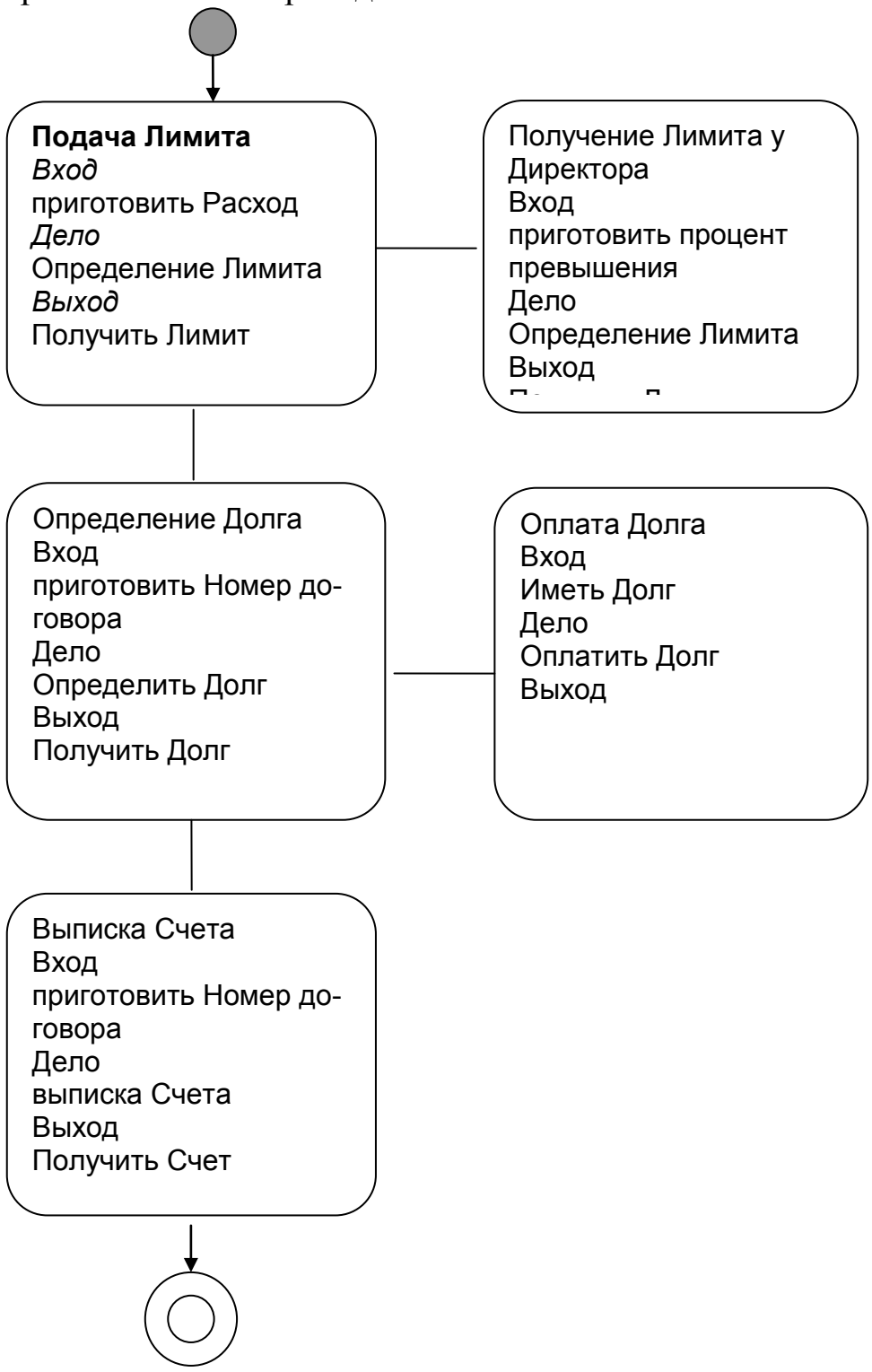
Приклад «Діаграма станів об'єктів системи Банкомат»:



Приклад «Побудова діаграми «Станів і переходів»

Використовуючи методологію об'єктного аналізу системи «Приватний підприємець», побудуємо діаграму «Станів і переходів» об'єкту «ЧаП» прецеденту «Подача відомостей до енергонагляду».

Діаграма «Станів і переходів»:



3.9 Приклад 06 «Аналіз «Станів системи»

Діаграма «Станів системи» (SateSystem)

Економічні системи дискретні. Існує хоч скільки завгодно малий проміжок часу, протягом якого об'єкти системи пасивні – активність одного об'єкту системи вже закінчилася, а активність іншого об'єкту ще не почалася.

Система переходить в новий стан у результаті виконання об'єктом дії у відповідь на деяку подію.

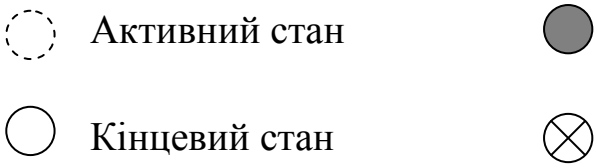
Результат реакції об'єкту на подію є новий стан системи.

Подія є те, що викликає зміну стану об'єкту – об'єкт набув нового значення властивості, об'єкт реалізував свою функціональність (поведінка), об'єкт обробив подію, об'єкт ініціював подію, об'єкт змінив значення властивості іншого об'єкту і тому подібне.

Діаграма станів системи – це графічне представлення проекції системи в просторі її станів. Використовуються для побудови математичних моделей станів системи й сценаріїв прецедентів.

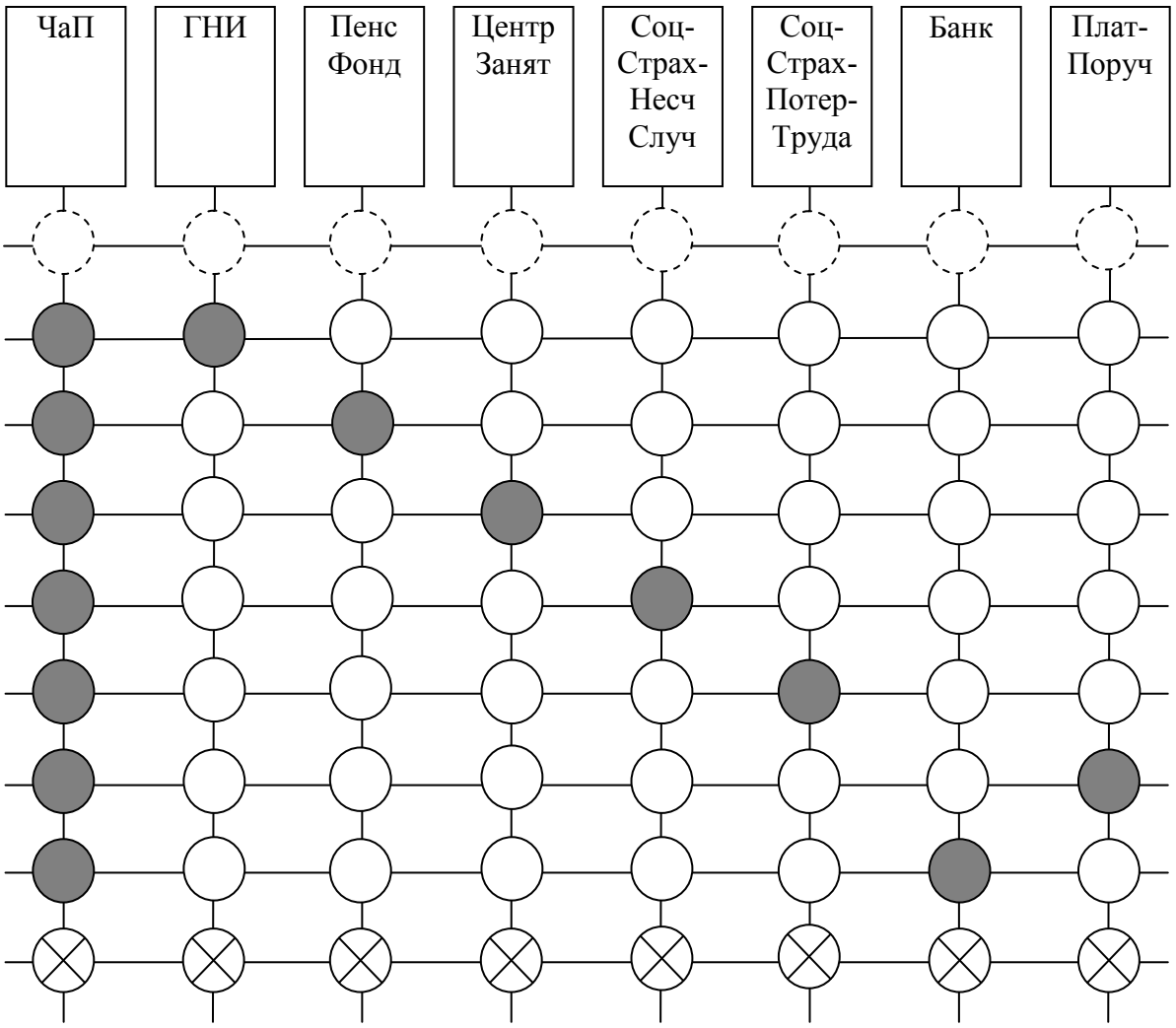
Позначення
Перетин:

Лінія життя об'єкту

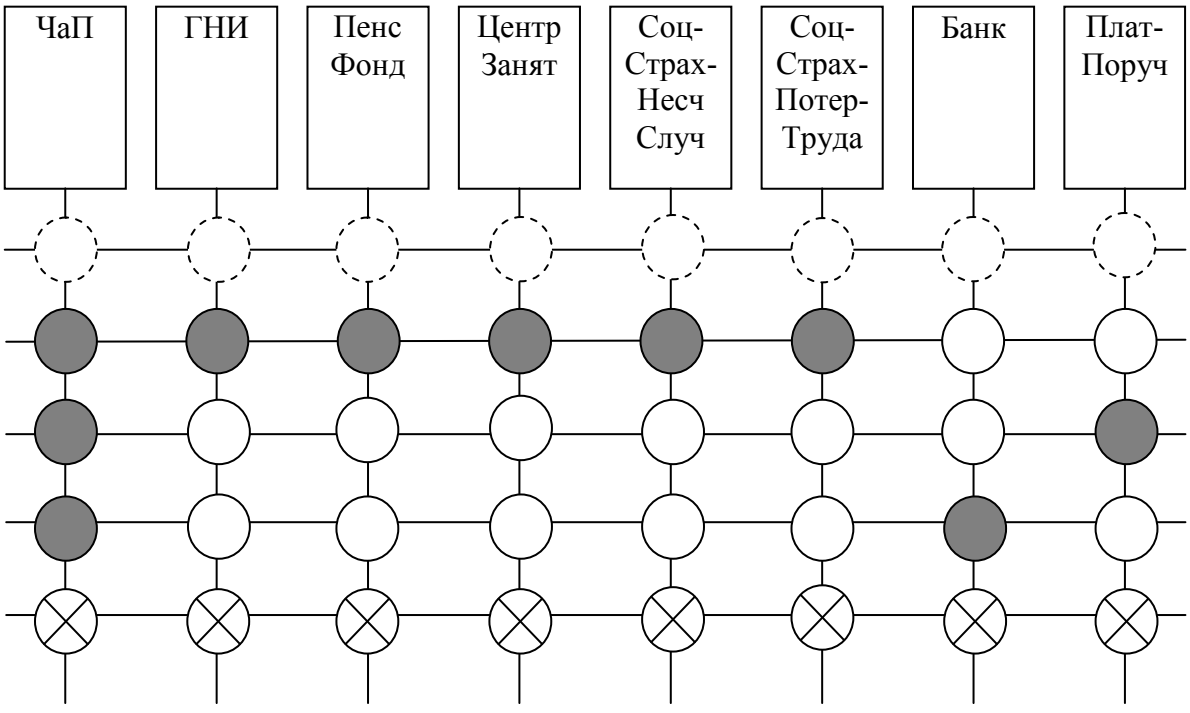


Нижче наведено приклад діаграми станів основного сценарію прецеденту «Ежемесячные отчисления» системи «Частный предприниматель». Показано дві діаграми станів системи – одна для випадку реального сценарію, а друга – з урахуванням особливостей програмної реалізації системи.

Реальне представлення системи



Віртуальне представлення системи



3.10 Приклад 07 «Аналіз кооперації об'єктів»

Мета кооперації «... є в тому, щоб специфікувати особливості реалізації окремих варіантів використання або окремих значущих операцій в системі» [1, с.170].

Діаграми «Кооперації» (Collaboration) використовуються для відображення взаємодії між об'єктами на основі повідомлень в контексті статичної структури системи й призначені для специфікації структурних аспектів взаємодії. Особливість діаграм кооперації полягає в можливості графічно представляти структурні стосунки між об'єктами.

Кооперацію розглядають двох видів:
– кооперації рівня специфікації (використовуються рідко);
– кооперації рівня екземплярів (використовуються часто).

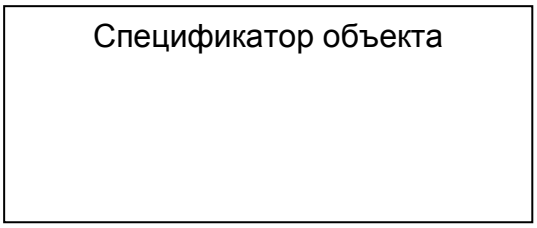
Кооперація рівня специфікації.
Мета – показати роль класифікаторів і роль асоціацій в даній взаємодії.

Кооперація рівня екземплярів
Кооперація рівня екземплярів раніше називалася діаграма об'єктів.
Мета – це показати об'єкти, зв'язки й повідомлення між ними в даній взаємодії. Відображають міжоб'єктний потік подій сценарію варіанта використання.

Позначення:

Об'єкт
Об'єкт – це екземпляр класу

Пасивний об'єкт той, який тільки обробляє дані або ще ініціює повідомлення, які вимагають обробки іншими об'єктами.



Специфікатор об'єкту – це рядок наступного формату:

ИмяОбъекта:/ИмяРолиОбъекта:ИмяКлассаОбъекта

або

ИмяОбъекта:ИмяКлассаОбъекта

або
:ИмяКлассаОбъекта

або
ИмяОбъекта:

або
ИмяОбъекта

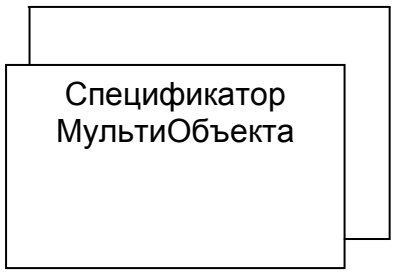
або
/ИмяРолиОбъекта:ИмяКлассаОбъекта

або
ИмяОбъекта:/ИмяРолиОбъекта

або
/ИмяРолиОбъекта

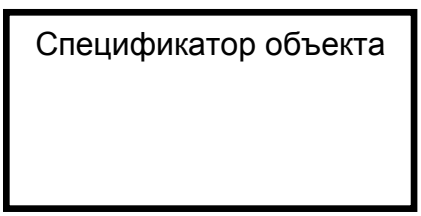
Мультіоб'єкт
Мультіоб'єкт – це множина екземплярів класу.

Позначення:



Активний об'єкт
Активний об'єкт – це об'єкт, який може ініціювати управління іншими об'єктами на основі Потоку або Нитки. Потік відрізняється від Нитки тим, що вимагає незрівнянно великих ресурсів.

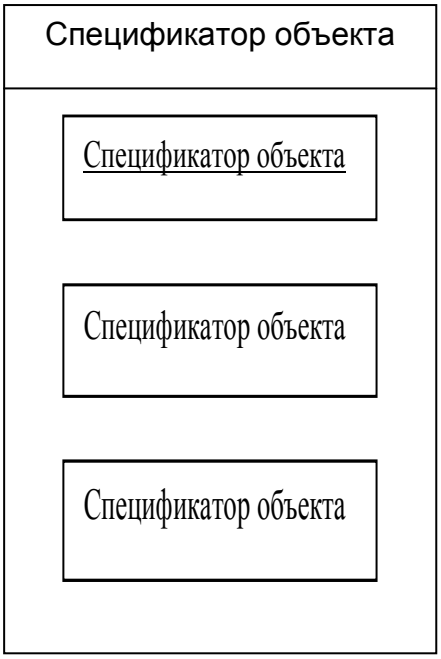
Позначення:



Складений об'єкт

Складений об'єкт (Composite Object) або об'єкт-композит призначений для представлення об'єкту, що має власну структуру й внутрішні потоки (нитки) управління. Це об'єкт, створений на основі класу-композиту.

Позначення:



Зв'язок

Позначення:

Зв'язок може мати деякий стереотип, який записується поряд з одним з її кінців і вказує на особливість зв'язку.

Стереотипи зв'язків:

<<association>>
Сполучає асоційовані об'єкти.

<<parameter>>
Об'єкт-джерело є параметром операції.

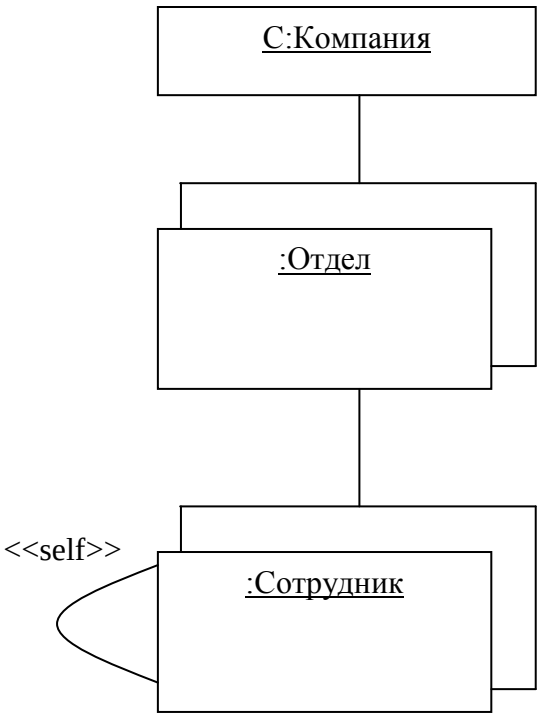
<<local>>
Об'єкт – джерело має локальну видимість.

<<global>>
Об'єкт – джерело має глобальну видимість.

<<self>>

Об'єкт сам собі посилає повідомлення.

Приклад «Види зв'язків»:



Повідомлення

Повідомлення використовується для специфікації зв'язку між об'єктами.

Направлено від об'єкту-джерела до об'єкту-одержувача.

Не тільки передає одержувачеві інформацію, але, можливо, і припускає деяку дію одержувача.

Синхронне повідомлення

Об'єкт-джерело чекає закінчення операції об'єкта-одержувача.

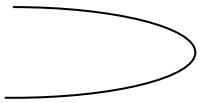
Асинхронне повідомлення

Об'єкт-джерело не чекає закінчення операції об'єкта-одержувача.

Поворотне повідомлення

Об'єкт-джерело отримує повідомлення про закінчення операції об'єкта-одержувача.

Зв'язок рефлексії



Об'єкт-джерело генерує повідомлення собі ж.

Стереотипи повідомлень.

<<call>>
Викликає роботу операції «ОбъектаПолучателя».

<<return>>
Повертає «ОбъектуИсточнику»результати роботи операції «ОбъектаПолучателя».

<<create>>
Вимагає створення об'єкту для виконання операції.

<<destroy>>
Вимагає знищення об'єкту.

<<send>>
Асихронно викликає роботу операції «ОбъектаПолучателя».

Формати запису повідомлень
Повідомлення може бути помічене уточнюючим рядком певного формату.

Розглядають також і такі повідомлення:

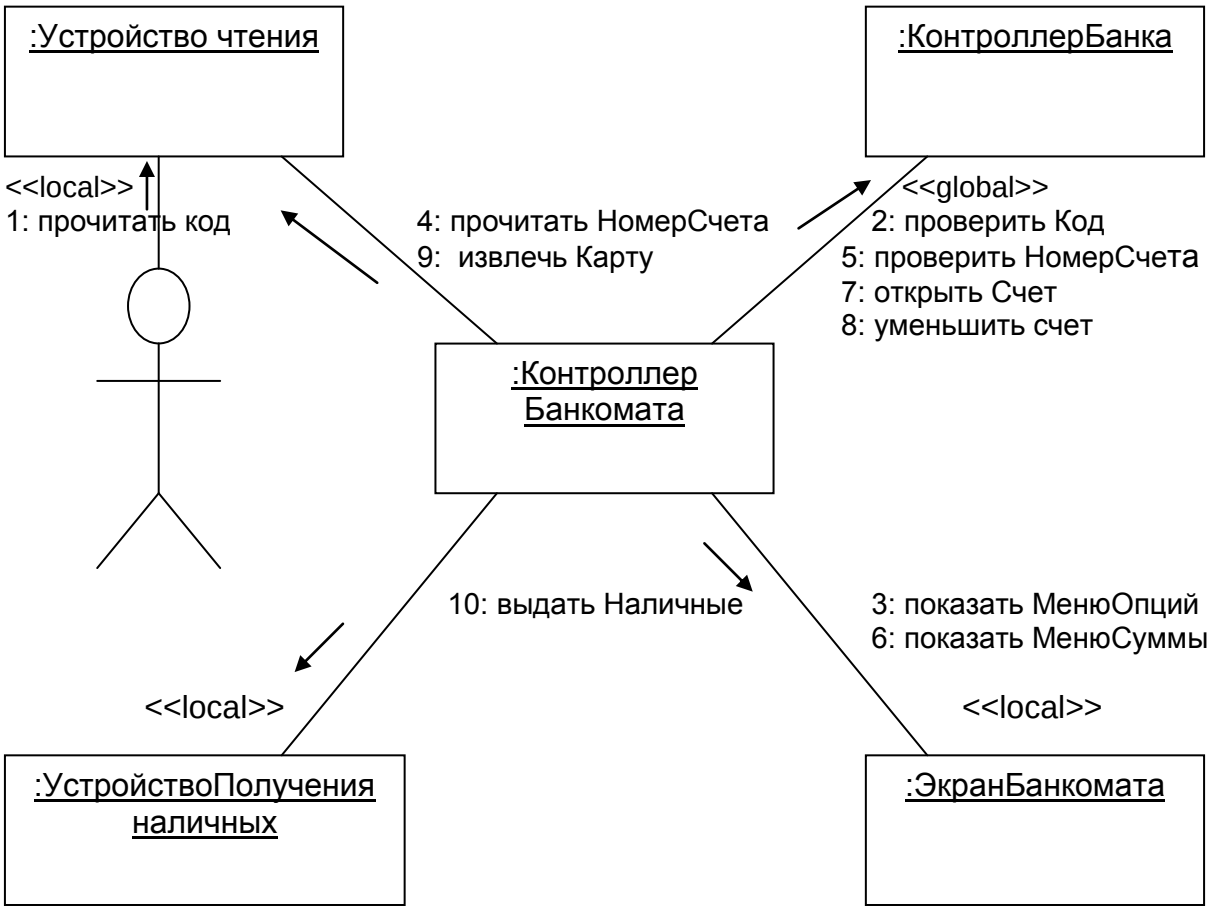
1. Повідомлення (Message)
Повідомлення (Message) – це засіб, яким «ОбъектИсточник» запрошує у «ОбъектаПолучателя» виконання його операції.

2. Інформаційне повідомлення (Informative)
Інформаційне повідомлення (Informative) – це засіб, яким «Объект-Получатель» забезпечується інформацією для зміни свого стану.

3. Повідомлення-запит (Interrogative)
Повідомлення-запит (Interrogative) –це засіб, яким запрошується інформація про «ОбъектаПолучателя».

4. Імперативне повідомлення (Imperative)
Імперативне повідомлення (Imperative) – це засіб, яким запрошується виконання дії «ОбъектаПолучателя».

Приклад «Діаграма кооперації системи управління банкоматом»:



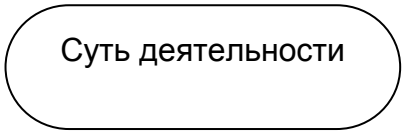
3.11 Приклад 08 «Аналіз діяльності»

Діаграма діяльності (Activity).

Діяльність – це сукупність елементарних дій, що реалізовують деякий алгоритм поведінки об'єкту.

На діаграмах діяльності відбивається логіка переходу від однієї діяльності до іншої при реалізації системою деякого сценарію.

Позначення:

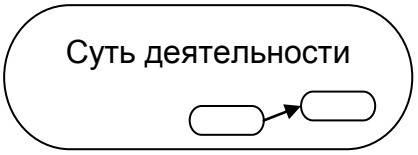


Суть діяльності може бути виражена вербально, кодом, псевдомовою.

Піддіяльність

Піддіяльність використовується для абстрагування від деталей піддіяльності.

Позначення:



Перехід

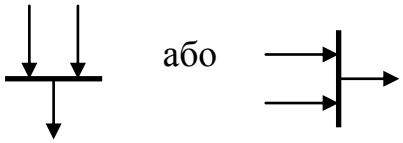
Перехід не є тригером, тобто спрацьовує відразу після закінчення діяльності або виконання відповідної дії стану.

Рішення

Рішення позначає вхід (зазвичай зверху або справа) і один або декілька виходів, помічених сторожовою умовою (булевим виразом).

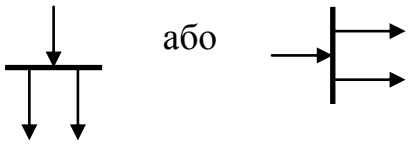
Злиття

Злиття має декілька вхідних переходів і один перехід, що виходить.



Розділення

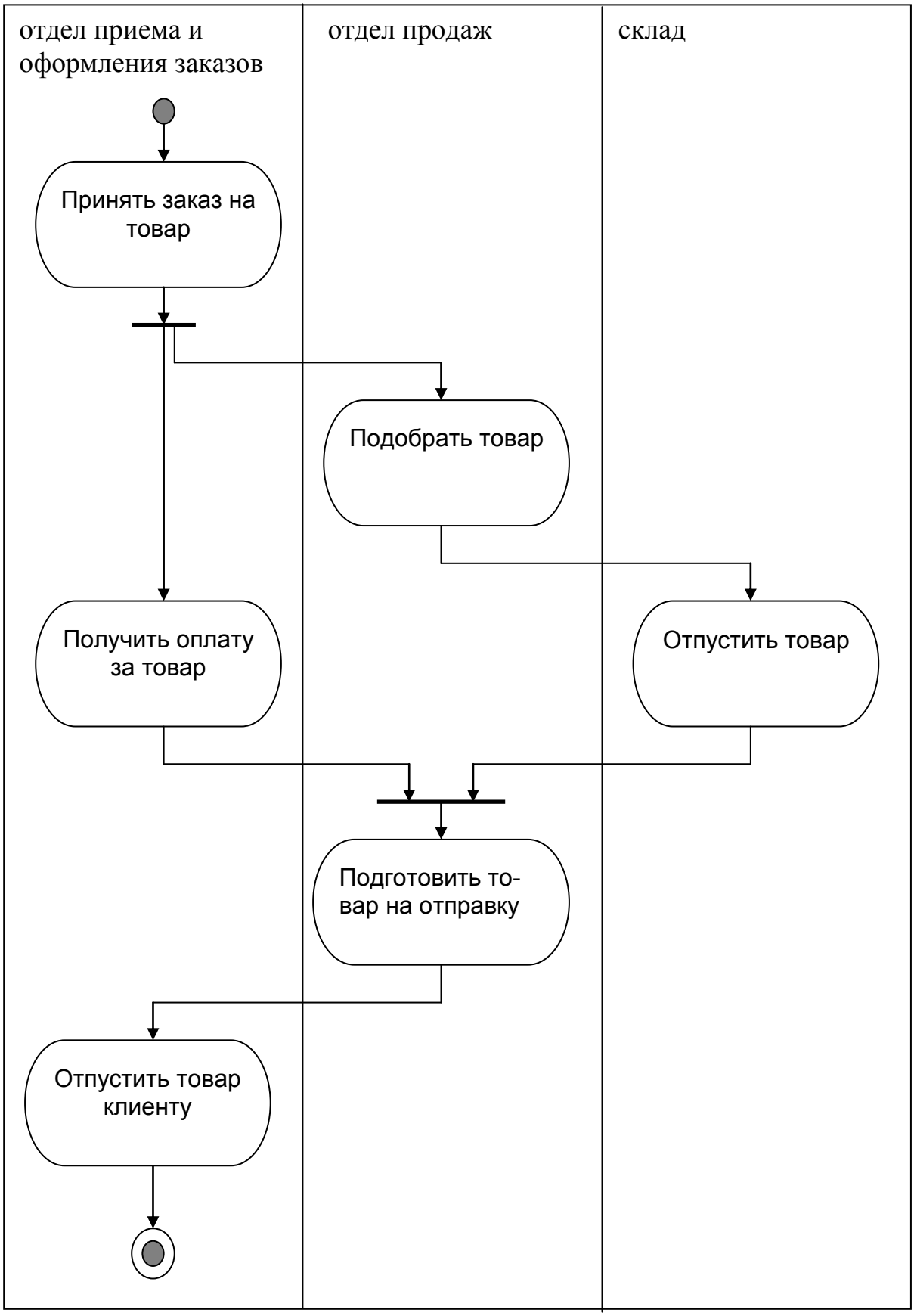
Розділення має один вхідний перехід і декілька переходів, що виходять.



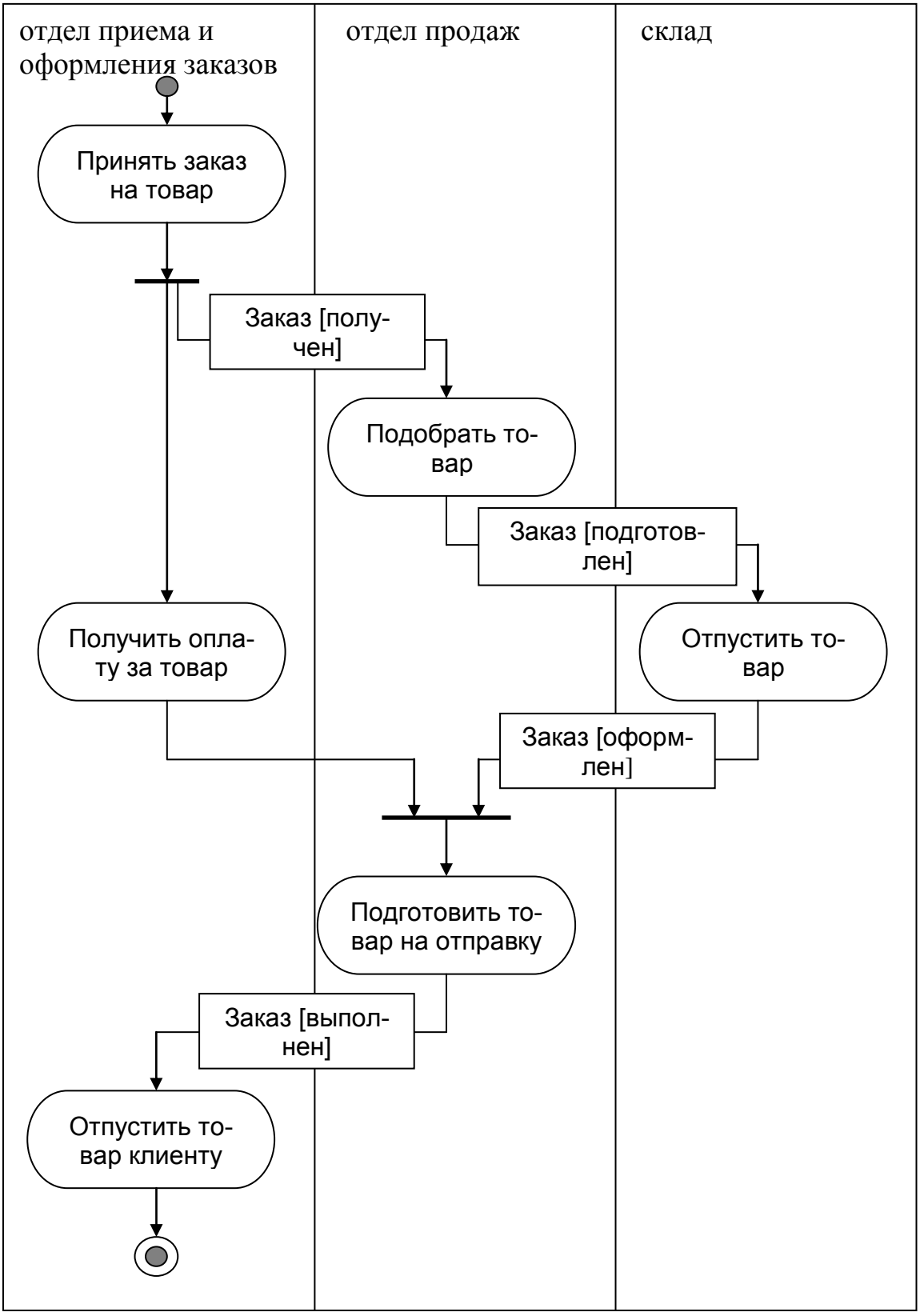
Доріжка

Діаграма діяльності може бути розбита на вертикальні доріжки. Доріжка може бути надана як структурному бізнес-підрозділу системи, так і об'єкту системи й в другому випадку графічні позначення об'єктів можуть перетинати лінії доріжок, показуючи цим, що перехід об'єкту до стану, відповідного наступній доріжці, припускає завершення стану, відповідного попередній доріжці.

Приклад 1 «Діаграма діяльності з бізнес-підрозділами»



Приклад 2 «Діаграма діяльності з об'єктами»



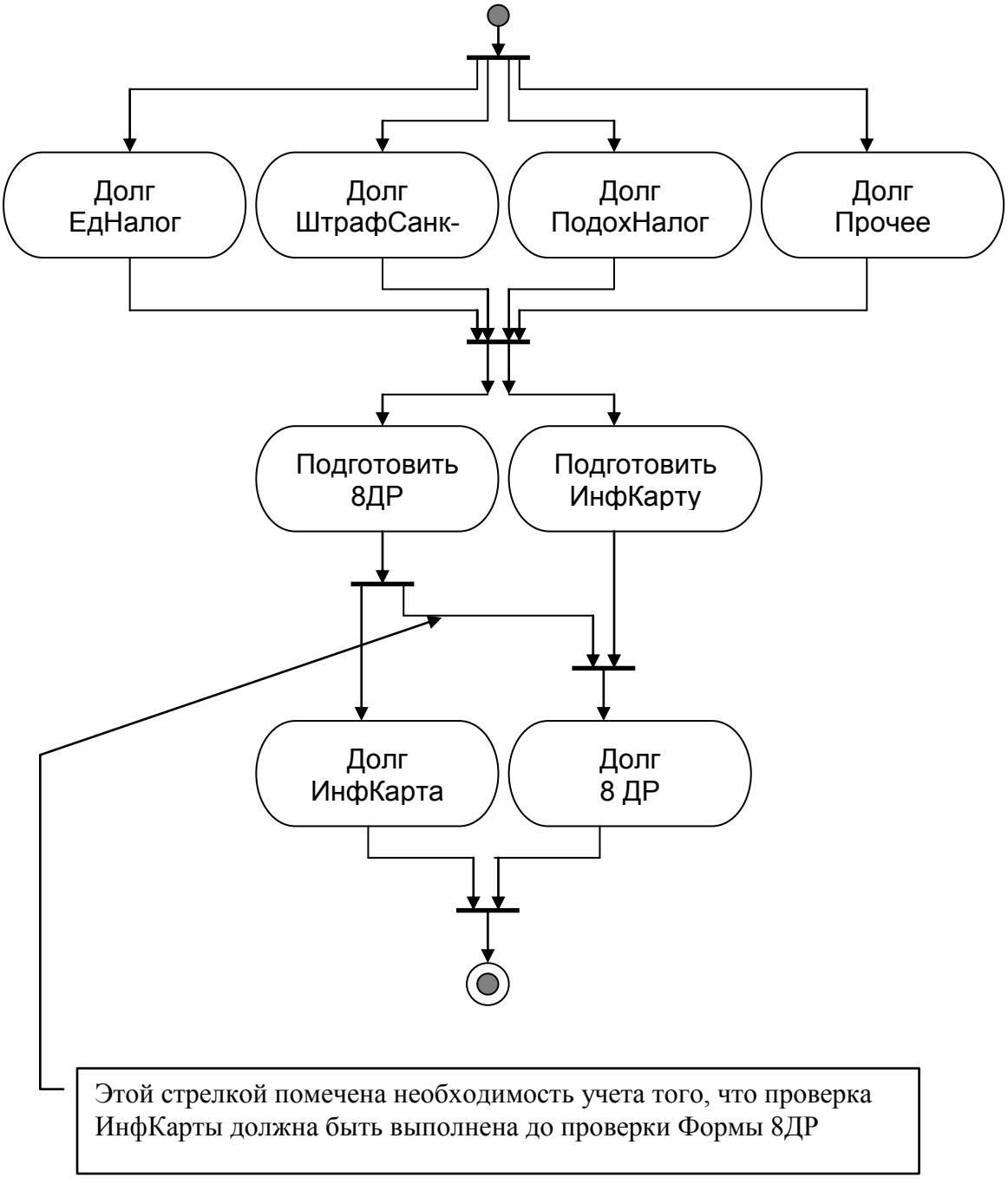
Синхронізація

Необхідність в синхронізації виникає, коли паралельно виконувані дії роблять вплив один на одного й вимагають узгодження.

Немає спеціального графічного примітиву для позначення узгодження у разі діаграм діяльності, а використовується позначення переходів «розділення», «злиття».

Приклад 3 «Діаграма діяльності з синхронізацією»

Прецедент «Квартальний звіт в ГНИ»



4 ОБ'ЄКТНЕ ПРОЕКТУВАННЯ СИСТЕМ

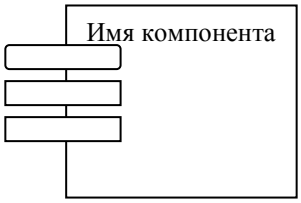
4.1 Приклад 09 «Аналіз модулів»

Діаграма Компонентів (Component)

Використовується для відображення матеріальної суті моделі системи. Встановлює залежності між програмними компонентами – початковим кодом, бінарним кодом, «исполнимым» кодом, яким часто відповідає файл модуля або компоненту. Основними графічними елементами діаграм компонентів є компоненти, інтерфейси й зв'язки між ними.

У разі бізнес-систем під компонентом слід розуміти організаційні підрозділи (відділи, служби), посади (менеджер, касир) або документи (рахунок, замовлення, накладна) і тому подібне.

Позначення:



Ім'я компоненту

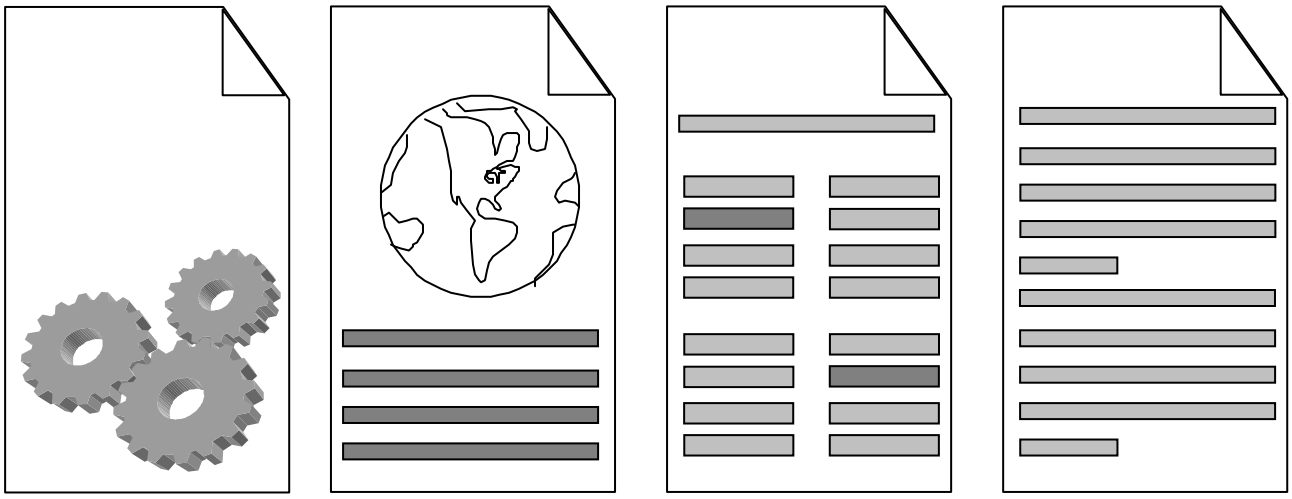
Записується усередині значка. Має формат:

<Имя компонента : Имя типа>

Як імена компонентів використовують імена файлів вигляду: exe, DLL, HTML, txt, doc, hlp, mdb, db та ін.

Графічні стереотипи компонента

Библиотека Веб файл Файл справки Файл кода



Графічні стереотипи специфікації виду компоненту дозволяють наочно підкреслити специфіку компоненту.

Текстові стереотипи компоненту

Це інший спосіб специфікації виду компоненту. Перед ім'ям компоненту ставлять слово специфікатор:

– <<file>> довільний файл;
– <<executable>> здійснимий файл;
– <<document>> текстовий файл;
– <<library>> бібліотека;
– <<source>> файл початкового коду;
– <<table>> таблиця бази даних.

Інтерфейс

Використовується для специфікації взаємодії користувача з системою, а також для позначення можливості зміни однієї частини системи без зміни інших її частин.

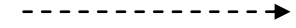
Позначення:

Або

Відношення

Залежність

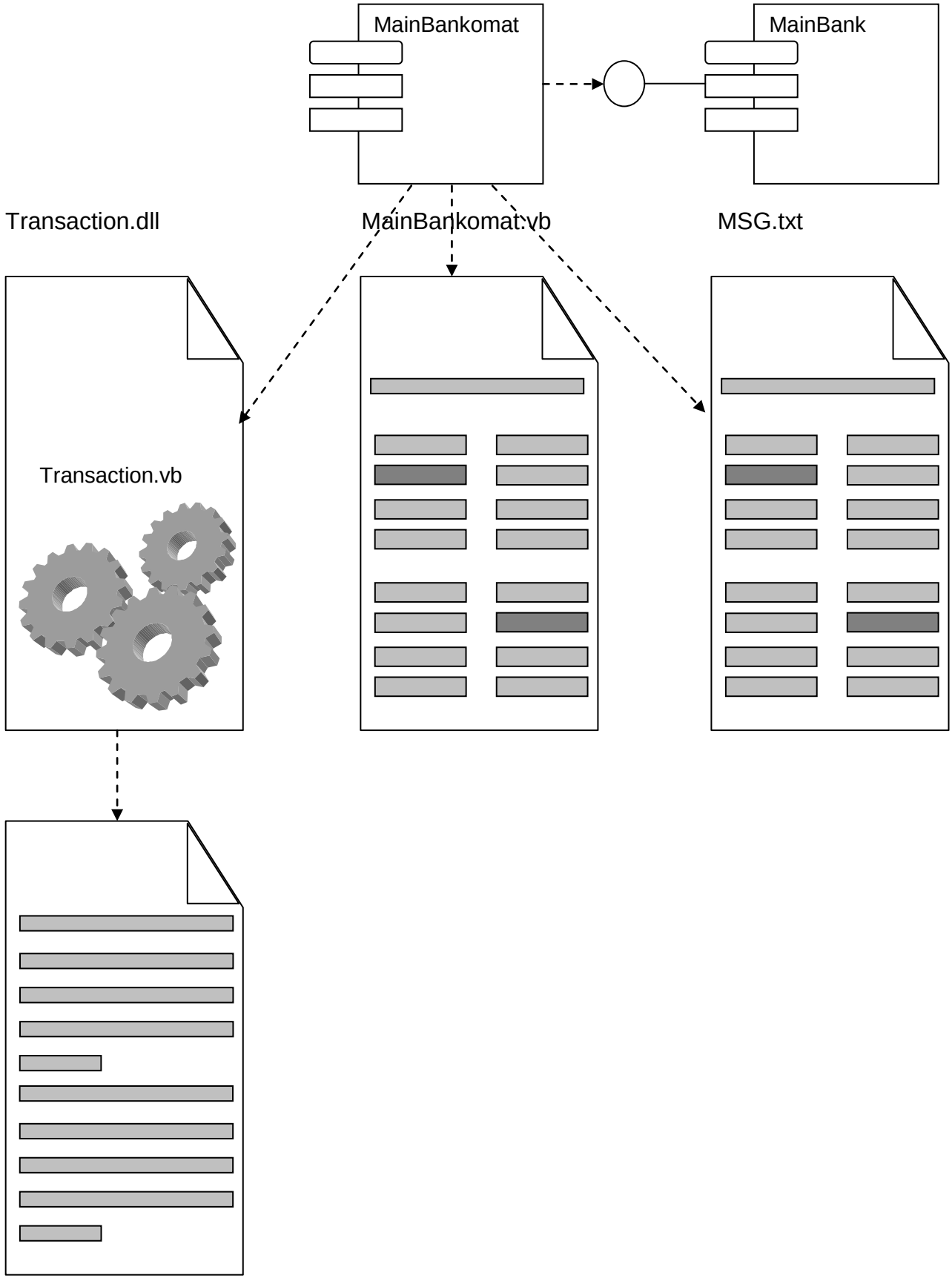
Якщо зміна одного компоненту надає вплив на роботу іншого компоненту, то говорять про відношення залежності й зображають це пунктирною лінією зі стрілкою, направленою від залежного компоненту (імпортер) до того, який робить вплив.



Реалізація

Якщо один компонент використовує елементи іншого компоненту (експортер), то говорять про відношення реалізації й зображають це суцільною лінією без стрілки.

Приклад «Діаграма компонентів системи Банкомат»



4.2 Приклад 10 «Аналіз топології»

Аналіз топології продовжує рішення питання фізичного представлення системи й на додаток до аналізу модулів використовується для розгортання системи між процесорами й пристроями у фізичному уявленні. Показує наявність фізичних з'єднань – шляхів передачі інформації між вузлами системи. Діаграма розгортання містить графічні зображення процесорів, пристроїв, процесів і зв'язків між ними.

Вузол

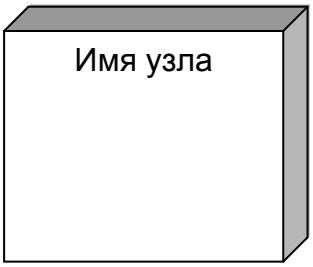
Вузол – це матеріальне представлення фізичної суті системи.

Текстові стереотипи вузлів:

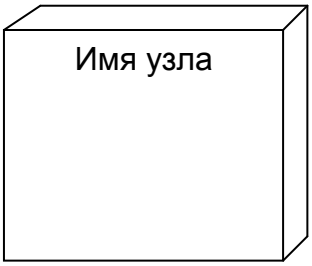
– <<processor>> процесор;
– <<sensor>> датчик;
<<modem>> модем;
– <<net>> мережа;
– <<printer>> принтер.

Графічні стереотипи вузлів:

Вузол – виконавець коду



Вузол – деякий пристрій



Відношення

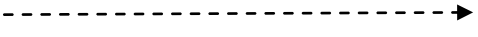
Відношення – це зв'язки між вузлами системи.

Асоціація

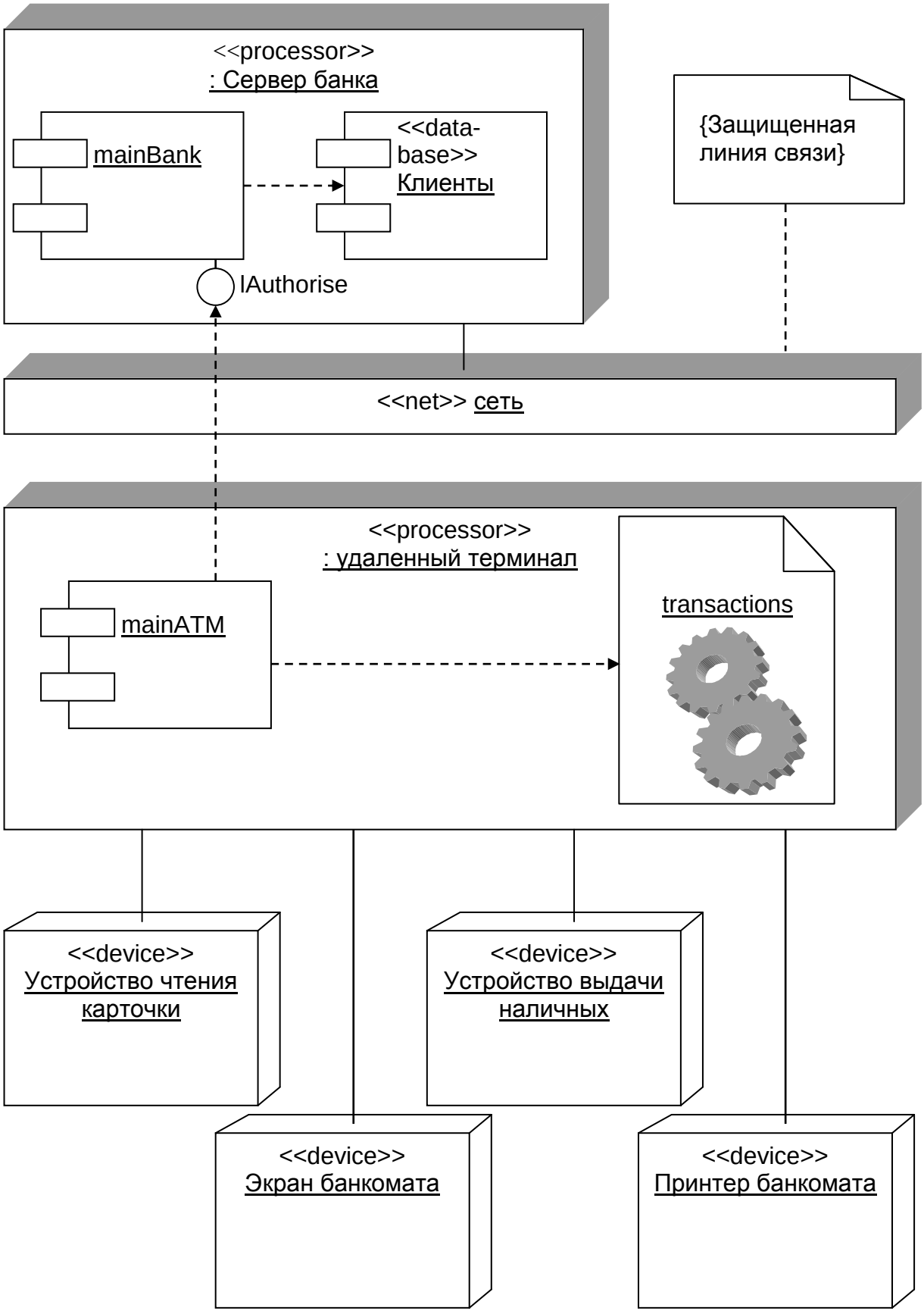
Асоціація – це зв'язок між рівноправними вузлами.

Залежність

Залежність – це зв'язок між вузлами, коли один впливає на роботу іншого. Стрілка вказує на впливаючий вузол.



Приклад діаграми «Розгортання»



4.3 Приклад 11 «Визначення специфікацій класів»

Побудова специфікацій класів – це процес документування атрибутів класів, метою якого є підготовка до кодування програмної моделі системи.

У прикладі «ЧаП» специфікація класу «ChaP» матиме вигляд:

Клас ChaP

Поля
mName
Закрите поле властивості Name. String типу.

Властивості
Name
Имя класса. ReadOnly. String типу.
ENLimit
Кількість електроенергії, яку можна було витрачати в звітному періоді.
ReadWrite. Single типу.
ENDolg
Кількість грошей, які треба було заплатити за електроенергію у минулому звітному періоді.
ReadWrite. Decimal типу.
ENCount
Кількість грошей, які треба заплатити за електроенергію в поточному звітному періоді.
ReadWrite. Decimal типу.

Методи
MainChaP(ByVal D As Date)
Використовується для перевірки умов і генерації події.
D – вхідний параметр – дата

Події
evENLimit
Викликає метод визначення Ліміту об'єкту ENLimit
Аналогічно визначаються специфікації інших класів.

Після визначення специфікацій класів будується перший реліз програмної моделі.

4.4 Приклад 12 «Тестування потоку подій»

Тестування потоку подій проводиться відповідно діаграмам послідовності подій виявлених сценаріїв системи. Наприклад, в завданні «Квартальний звіт в податковій службі» тест потоку подій може мати вигляд, приведений нижче.

Тест 1

Якщо

Будь-який день до 1 числа місяця, наступного за звітним кварталом.

то очікується відповідь:

```
MsgBox("День --  " & myD & "  -- начался!")
MsgBox("День --  " & myD & "  -- закончился!")
```

Тест 2

Якщо

Перше число місяця, наступного за звітним кварталом.

то очікується відповідь:

```
MsgBox("День --  " & myD & "  -- начался!")"Обр-ся
соб
  Code"
MsgBox("Обработано событие Бегунок от ЧаП
  для Инспектор")
MsgBox("Обработано событие ДолгЕдНалог от ЧаП
  для ДогЕдНалог")
MsgBox("Обработано событие ДолгПодох от ЧаП
  для ДолгПодох")
MsgBox("Обработано событие ДолгШтраф от ЧаП
  для ДолгШтраф")
MsgBox("Обработано событие СделатьФорму8ДР от ЧаП
  для СделатьФорму8ДР")
MsgBox("Обработано событие ИнфКарта от ЧаП
```

```
  для ИнфКарта")
MsgBox("Обработано событие Форма8ДР от ЧаП
  для Форма8ДР")
MsgBox("Обработано событие Форма1ПП от ЧаП
  для Форма1ПП")
MsgBox("Обработано событие Расчет от ЧаП для Расчет")
MsgBox("День --  " & myD & "  -- закончился!")"Обр-ся
  Money"
```

Тест 3

Якщо

Будь-який день після 1 числа місяця, наступного за звітним кварталом.

то очікується відповідь:

```
MsgBox("День - " & myD & " - начался!")
MsgBox("День - " & myD & " - закончился!")
```

Тест 4

Якщо

Перше число місяця, наступного за звітним кварталом.
Є ДолгЕдЕалог.

то очікується відповідь:

```
MsgBox("День - " & myD & " - начался!")
  "Обр-ся соб-е Code"
MsgBox("Обработано событие Бегунок от ЧаП
  для Инспектор")
MsgBox("Долг ДолгЕдНалог устранен")
MsgBox("Обработано событие ДолгЕдНалог от ЧаП
  для ДогЕдНалог")
MsgBox("Обработано событие ДолгПодох от ЧаП
  для ДолгПодох")
MsgBox("Обработано событие ДолгШтраф от ЧаП
```

```
  Для ДолгШтраф")
MsgBox("Обработано событие СделатьФорму8ДР от ЧаП
  для СделатьФорму8ДР")
MsgBox("Обработано событие ИнфКарта от ЧаП
  для ИнфКарта")
MsgBox("Обработано событие Форма8ДР от ЧаП
  для Форма8ДР")
MsgBox("Обработано событие Форма1ПП от ЧаП
  для Форма1ПП")
MsgBox("Обработано событие Расчет от ЧаП для Расчет")
MsgBox("День - " & myD & " - закончился!")
  "Обр-ся соб-е Money"
```

Тест 5

Якщо

Перше число місяця, наступного за звітним кварталом.
Є ДолгПодох

то очікується відповідь:

```
MsgBox("День   " & myD & " - начался!")
  "Обр-ся соб-е Code"
MsgBox("Обработано событие Бегунок от ЧаП
  для Инспектор")
MsgBox("Обработано событие ДолгЕдНалог от ЧаП
  для ДогЕдНалог")
MsgBox("Долг ДолгПодох устранен")
MsgBox("Обработано событие ДолгПодох от ЧаП
  для ДолгПодох")
MsgBox("Обработано событие ДолгПодох от ЧаП
  для ДолгПодох")
MsgBox("Обработано событие СделатьФорму8ДР от ЧаП
  для СделатьФорму8ДР")
```

```
MsgBox("Обработано событие ИнфКарта от ЧаП
  для ИнфКарта")
MsgBox("Обработано событие Форма8ДР от ЧаП
  для Форма8ДР")
MsgBox("Обработано событие Форма1ПП от ЧаП
  для Форма1ПП")
MsgBox("Обработано событие Расчет от ЧаП
для Расчет")
```

Тест 6

Якщо

Залежність число місяця, наступного за звітним кварталом.
Є ДолгРасчет

то очікується відповідь:

```
MsgBox("День - " & myD & " - начался!")
  "Обр-ся соб-е Code"
MsgBox("Обработано событие Бегунок от ЧаП
  для Инспектор")
MsgBox("Обработано событие ДолгЕдНалог от ЧаП
  для ДогЕдНалог")
MsgBox("Обработано событие ДолгПодох от ЧаП
  для ДолгПодох")
MsgBox("Обработано событие ДолгПодох от ЧаП
  для ДолгПодох")
MsgBox("Обработано событие СделатьФорму8ДР от ЧаП
  для СделатьФорму8ДР")
MsgBox("Обработано событие ИнфКарта от ЧаП
  для ИнфКарта")
MsgBox("Обработано событие Форма8ДР от ЧаП
  для Форма8ДР")
```

4.5 Приклад 13 «Тестування діяльності об'єктів»

Тестувати, як правило, слідує граничні набори даних, «типовий», а також критичні в деякому розумінні набори даних.

Тестування алгоритму слід проводити по наступній схемі.
Якщо (приводитися набір початкових даних).
Тоді очікується відповідь: (наводиться результат ручного прорахунку).
Дійсно: (наводиться результат комп'ютерного прорахунку).

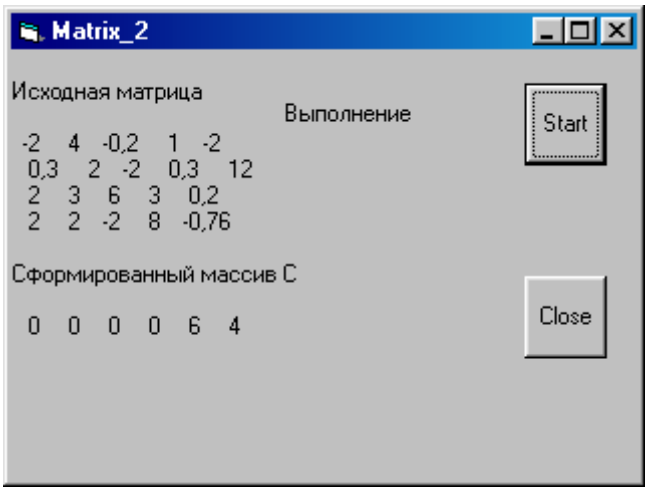
Збіг очікуваної відповіді й отриманого в результаті комп'ютерного прогону свідчить про правильність побудованого алгоритму.

Нижче, в прикладі «Старий знайомий» було надано масив *a(m,n)* цілих чисел. Для кожної трійки якого треба було знайти максимальний елемент з тих, що утворюють підматрицю матриці А так, що знайдений елемент є правим нижнім кутом підматриці. Знайдені максимальні елементи скласти в масив З, заповнюючи його з кінця. Фрагмент рішення, а саме, тестування алгоритму як функціональності деякого об'єкту системи, показаний нижче.

Тест 1
Якщо

$$a = \begin{pmatrix} -2 & 4 & -0,2 & 1 & -2 \\ 0,3 & 2 & -2 & 0,3 & 12 \\ 2 & 3 & 6 & 3 & 0,2 \\ 2 & 2 & -2 & 8 & -0,76 \end{pmatrix},$$

Тоді очікується відповідь: $C = \{0,0,0,0,6,4\}$.
Дійсно:



СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++ / Г. Буч. – 2-е изд. пер. с англ. – М.: Издательство Бином, СПб.: Невский диалект, 2000. – 560 с.

2. Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя. 2-е изд.: пер. с англ. Мухин Н. / Г. Буч, Д. Рамбо, И. Якобсон. – М.: ДМК Пресс. – 2006. – 496 с.: ил.

3. Леоненков А. В. Самоучитель UML / А. В. Леоненков. – 2-е изд., перераб. и доп. – СПб.: БХВ – Петербург, 2004. – 432 с.: ил.

4. Эдвард Й. Объектно-ориентированный анализ и проектирование систем / Й. Эдвард, А. К. Аргила. – М.: Издательство Лори, 2007.– 284 с.

5. Ларман К. Применение UML 2.0 и шаблонов проектирования. / К. Ларман. – М.: Издательсктй дом Вильямс, 2007. – 736 с.

6. Трофимов С. А. CASE-технологии. Практическая работа в Rational Rose / С. А Трофимов – М.: Издательство БИНОМ, 2002. – 288 с.

7. CASE-технологии. Практикум. / Д. Э. Федотова, Ю. Д. Семенов, К. Н. Чижик. – М.: Издательство БИНОМ, 2005. –160 с.

8. Фаулер М. UML. Основы. Краткое руководство по унифицированному языку моделирования / М. Фаулер, С. Кендалл. – М.:Символ-Плюс, 2002. – 192 с.

9. Максимчук Р. А. UML для простых смертних / Р. А. Максимчук, Э. Д. Нейбург. – М.: Издательство Лори, 2008. – 304 с.

10. Путилин А. Б. Компонентное моделирование и программирование на языке UML: практическое руководство по проектированию информационно-измерительных систем / А. Б. Путилин, Е. А. Юрагов. – М.: Издательство НТ Пресс, 2005. – 664 с.

11. Киммел П. UML. Основы визуального анализа и проектирования / П. Киммел. – Издательская группа АСТ, 2008. – 272 с.

12. Боггс У. UMl и Rational Rose 2002 / У. Боггс, М. Боггс. – М.: Издательство Лори, 2004. – 509 с.