Міністерство освіти і науки України
Одеський національний морський університет
Кафедра кібербезпеки та захисту інформації

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни «Бізнес-процеси в кібербезпеці»
другого (магістерський) рівня вищої освіти
за спеціальністю F5 Кібербезпека та захист інформації
галузі знань F Інформаційні технології

Одеса, 2025

Міністерство освіти і науки України
Одеський національний морський університет
Кафедра кібербезпеки та захисту інформації

КОНСПЕКТ ЛЕКЦІЙ
з дисципліни «Бізнес-процеси в кібербезпеці»
другого (магістерський) рівня вищої освіти
за спеціальністю F5 Кібербезпека та захист інформації
галузі знань F Інформаційні технології

Затверджено
на засіданні кафедри КБЗІ
Протокол № 2 від 06.10.2025 р.

Одеса, 2025

Конспект лекцій з дисципліни «**Бізнес-процеси в кібербезпеці**» для другого (магістерський) рівня вищої освіти за спеціальністю F5 Кібербезпека та захист інформації галузі знань F Інформаційні технології / Укл.: *Лебедєва О.Ю.* – Одеса, 2025. – 107 с.

Укладач: **Лебедєва О.Ю.**, к.т.н., доц.

ЗМІСТ

Змістовий модуль 1. Методології моделювання бізнес-процесів 5
Лекція 1. Поняття бізнес процесу. Способи опису 5
1. Основні поняття 5
2. Способи опису бізнес-процесів 6
3. Методологія SADT 7
Лекція 2. Нотація IDEF0 11
1. Компоненти синтаксису IDEF0 11
2. Контекстна діаграма IDEF0 12
3. Приклади використання нотації IDEF0 18
Лекція 3. Нотація IDEF3 23
1. Компоненти синтаксису IDEF3 23
2. Діаграма PFDD 24
Лекція 4. Нотація IDEF3 (продовження) 33
1. Діаграма OSTN 33
2. Приклади використання нотації IDEF3 35
Лекція 5. Нотація IDEF1X 37
1. Компоненти синтаксису IDEF1X 37
2. Діаграма IDEF1X 37
Лекція 6. Нотація UML. Структурне моделювання 45
1. Модель класів 45
2. Компоненти синтаксису моделі класів 45
Лекція 7. Нотація UML. Моделювання поведінки 54
1. Моделі стану та взаємодії 54
2. Діаграми станів 60
Лекція 8. НОТАЦІЯ BPMN 67
1. Ролі або зони відповідальності 67
2. Об'єкти потоку управління 69
Лекція 9. Нотація DFD 83
1. Компоненти синтаксису DFD 83
2. Варіанти графічних нотацій 84
2. Приклади використання DFD 88
Лекція 10. Нотація EPC 91
1. Компоненти синтаксису EPC 91
2. Приклади використання DFD 104
Література 107

Змістовий модуль 1. Методології моделювання бізнес-процесів
Лекція 1. Поняття бізнес процесу. Способи опису

1. Основні поняття
2. Способи опису бізнес-процесів
3. Методологія SADT

1. Основні поняття

Бізнес-процес – послідовність дій (підпроцесів), спрямована на отримання заданого результату.

Бізнес-процес являє собою систему послідовних, цілеспрямованих і регламентованих видів діяльності, в якій за допомогою керуючого впливу і за допомогою ресурсів входи процесу перетворюються в виходи, результати процесу, що представляють цінність для споживачів.

Ключовими властивостями бізнес-процесу є те, що це кінцева і взаємопов'язана сукупність дій, що визначається відносинами, мотивами, обмеженнями і ресурсами всередині кінцевої множини суб'єктів і об'єктів, які об'єднуються в систему заради спільних інтересів з метою отримання конкретного результату, відчужуваного або споживаного самою системою.

Бізнес-процеси поділяють на:

– основні,
– супутні,
– допоміжні,
– які забезпечують,
– процеси управління,
– процеси розвитку.

Основними бізнес-процесами є процеси, орієнтовані на виробництво товарів або надання послуги, є цільовими об'єктами створення підприємства і забезпечують отримання заданого результату господарської діяльності.

Супутні бізнес-процеси – це процеси, в результаті яких формуються супутні результати господарської діяльності підприємства.

Допоміжні та які забезпечують бізнес-процеси – це процеси, призначені для життєзабезпечення основних і супутніх процесів і орієнтовані на підтримку їх специфічних рис.

Бізнес-процеси управління – це процеси, що охоплюють увесь комплекс функцій управління на рівні кожного бізнес-процесу і підприємства в цілому.

Бізнес-процесами розвитку є процеси вдосконалення виробленого товару або послуги, процеси розвитку технологій, процеси модифікації устаткування, а також інноваційні процеси.

Опис бізнес-процесів проводиться з метою їх подальшого аналізу і поліпшення.

Моделювання бізнес-процесів дозволяє проаналізувати не тільки як працює підприємство в цілому, як воно взаємодіє зі зовнішніми організаціями, замовниками та постачальниками, а й як організована діяльність на кожному окремо взятому підрозділі, ділянці, робочому місці.

Поняття бізнес-процес лежить в основі процесного підходу до аналізу і синтезу діяльності організації. *Процесний підхід* дозволяє розглядати діяльність організації як пов'язану систему бізнес-процесів, кожен з яких протікає у взаємозв'язку з іншими бізнес-процесами або зовнішнім середовищем.

2. Способи опису бізнес-процесів

Опис бізнес-процесів організації можуть вимагати різні стандарти, опис процесів має наступні переваги:

- У процесі опису з'ясовується справжня процедура протікання процесу, і з'ясовуються проблемні місця.
- Опис процесів має на увазі, що інформація про процес буде зафіксована і однаково зрозуміла для всіх учасників процесу.
- Опис процесу дозволяє чітко розмежувати сфери відповідальності між співробітниками.
- Описаний процес дозволяє побачити рух документів, товарних цінностей в організації.
- Описаний процес дозволяє визначити найбільш значущі ресурси для організації і ступінь їх використання.

Бізнес-моделювання – діяльність з виявлення, опису, аналізу існуючих бізнес-процесів, а також проектування нових бізнес-процесів.

Під *бізнес-моделлю* будемо розуміти структурований графічний опис мережі процесів і / або функцій / операцій, пов'язаних з даними, документами, організаційними одиницями та іншими об'єктами, що відображають існуючу або передбачувану діяльність організації.

Текстовий спосіб – такий спосіб, що являє собою простий текстовий послідовний опис бізнес-процесу. Наприклад: «Відділ продажів складає договір і погоджує його з юридичним відділом».

Багато підприємств розробили і використовують у своїй діяльності регламентуючі документи, частина яких є процесними регламентами і представляє не що інше, як текстовий опис бізнес-процесів.

Для цілей аналізу і оптимізації діяльності компанії даний варіант має суттєвий недолік. Опис бізнес-процесу в текстовому вигляді системно розглянути і проаналізувати фактично неможливо.

Табличний спосіб опису бізнес-процесу є більш формалізованим і передбачає розбиття бізнес-процесу по осередках структурованої таблиці, в якій кожен стовпець і рядок мають деякий певне значення.

Розглянемо приклад табличного опису бізнес-процесу. Нехай:

- Процес – закупка,
- Власник – заступник комерційного директора,
- Мета процесу – забезпечення потреби виробництва матеріалами та комплектуючими,
- Короткий опис процесу – організація забезпечення ТМЦ, зберігання і передача їх у виробництво, вибір та оцінки постачальників.

Тоді опис такого процесу табличним способом продемонстровано на рисунку 1.1.

№	Субпроцес	Зміст	Власник	Учасники
1.	Узагальнення та уточнення переліку ТМЦ, що закуповуються	Уточнення переліку ТМЦ, що закуповуються. Оформлення карток дозволу щодо заміни	Начальник відділу зовнішньої комплектації, відділ матеріально-технічного постачання	Відділ автоматизованого управління виробництвом
	Оцінка та вибір постачальників	Вибір альтернативних постачальників. Збір даних про	Заступник комерційного директора	Провідні інженери відділу

		постачальника. Оцінка постачальників. Порівняльний аналіз постачальників. Вибір постачальника		зовнішньої комплектації, відділ матеріально-технічного постачання
	…			

Рисунок 1.1 – Опис процесу табличним способом

Дану таблицю читати простіше, з неї легше зрозуміти, хто за що відповідає, в якій послідовності в бізнес-процесі виконуються роботи, і, відповідно, бізнес-процес простіше проаналізувати.

Таблична форма опису бізнес-процесів більш ефективна в порівнянні з текстовою, тому поширена і застосовується для опису бізнес-процесів в додатку до завдань їх автоматизації.

При використанні табличного способу на початковому етапі необхідно описати входи і виходи процесу (постачальників і споживачів), що управляють (внутрішні та зовнішні) і види ресурсів (людські і матеріальні). Для цього складається «Відомість визначення процесу».

Необхідно описати субпроцеси, а також види супровідної документації та ризики зриву процесу. Опис бізнес-процесу повинно містити блок-схему і логіку процесу.

В даний час найбільший розвиток і застосування при описі бізнес-процесів отримують графічні підходи і методи. Визнано, що вони мають найбільшу результативністю при вирішенні задач по опису, аналізу та раціоналізації діяльності підприємства.

Блок-схема включає процес, представлений у вигляді сутностей (прямокутників довільної форми), пов'язаних відносинами (стрілками), які задають послідовність виконання функцій процесу.

Блок-схема містить опис наступних атрибутів процесу:

– власник процесу,
– умови початку процесу,
– замовник процесу
– очікувані вихідні результати.

При отриманні вихідних даних відбуваються оцінка процесу і аналіз фактичних показників. Потім формуються вимоги до додаткових ресурсів для поліпшення діяльності процесу і визначаються види ресурсів.

Логіка процесу може бути представлена у вигляді таблиці, яка містить стовпці:

– дія,
– зміст
– відповідальний виконавець
– учасники.

Дії – це етапи процесу, причому кожному рядку таблиці відповідає свій етап.

Сукупність даних етапів визначає *алгоритм виконання процесу*, що описується у вигляді блок-схеми, де дії представлені у вигляді прямокутників, а умовні переходи – у вигляді ромбів.

Зміст дає уявлення про дії на кожному етапі процесу. Це можуть бути різні види документів для даного етапу, а також аналіз пройдених етапів (умова переходу), оцінки, запити та пояснення. Кожен етап (дію) має свого відповідального виконавця та учасників.

3. Методологія SADT

Методологія структурного аналізу і проектування (Structured Analysis and Design Technique, SADT) була створена на наприкінці 60-х рр. XX ст. Дугласом Россом. SADT

знайшла своє застосування в області опису великої кількості складних штучних систем з широкого спектру областей.

У 1973 р. вперше за допомогою SADT був реалізований великий аерокосмічний проект. К 1981 р. методологія SADT використовувалася більш ніж в 50 компаніях при роботі над проектами, що охоплювали різні проблемні області, в тому числі телефонні мережі, аерокосмічне виробництво, управління та контроль, облік матеріально-технічних ресурсів.

Метод SADT підтримується Міністерством оборони США, яке було ініціатором розробки сімейства стандартів IDEF (ICAM DEFinition), що є основною частиною програми ICAM (Integrated Computer Aided Manufacturing – інтегрована комп'ютеризація виробництва), що проводиться за ініціативою військово-повітряних сил США.

SADT-методологія – сукупність методів, правил та процедур, призначених для побудови функціональної структури складних ієрархічних систем у вигляді моделі, яка має дати відповідь на деякі наперед визначені питання. В основі цього методу моделювання систем лежить опис системи, створюваного за допомогою природної мови, що дозволяє вільно описати функціонування моделі, що моделюється. На основі графічних засобів SADT дескриптивний опис системи забезпечується зображенням її моделі, яке практично повністю усуває можливу неоднозначність семантичного опису. SADT – це методологія, розроблена спеціально для того, щоб полегшити опис та розуміння штучної системи середньої складності та її середовища до визначення вимог до програмного забезпечення або чогось іншого.

Застосування SADT методології засноване на формалізованому процесі створення системи при розбиття його на наступні фази:

- аналіз – визначення того, що система робитиме;
- проектування – визначення підсистем та їх взаємодія;
- реалізація – розробка підсистем окремо;
- об'єднання – з'єднання підсистем в єдине ціле;
- тестування – перевірка роботи системи;
- установка – введення системи в дію;
- функціонування – використання системи.

SADT-модель – це точний, повний і адекватний текстовий і графічний опис системи, що має конкретне призначення, виконане у вигляді ієрархічно організованої сукупності діаграм, створених на основі стандартного представлення даних. Це опис системи, у якої є єдиний суб'єкт, мета і одна точка зору за допомогою SADT-методології. Така модель являє собою сукупність ієрархічно впорядкованих та взаємопов'язаних діаграм, організованих у вигляді деревоподібної структури, де верхня діаграма є найбільш загальною, а найнижчі найбільш деталізовані.

У SADT-моделях використовуються як природна, так і графічна мови. Для передачі інформації про конкретну систему джерелом природної мови служать люди, що описують систему, а джерелом графічної мови - сама методологія SADT. Графічна мова SADT забезпечує структуру та точну передачу моделі семантики природної мови. Графічна мова SADT організує природну мову цілком певним і однозначним чином, завдяки чому SADT дозволяє описувати системи, які донедавна не піддавалися адекватному уявленню.

З точки зору SADT модель може бути зосереджена або на функціях системи або на її об'єктах. SADT-моделі, зорієнтовані на функції, прийнято називати *функціональними моделями*, а орієнтовані на об'єкти системи — *моделями даних*.

Згідно з авторами SADT процес моделювання, як процесу створення несуперечливої та корисної системи описів, складається з чотирьох послідовних етапів:

1. Збір інформації про досліджувану область.
2. Документування отриманої інформації.
3. Подання її у вигляді моделі.
4. Уточнення моделі у вигляді ітеративного рецензування.

Методологія SADT реалізовано в одному зі стандартів сімейства IDEF – IDEF0, який був затверджений в якості федерального стандарту США у 1993р.

Розглянемо основні елементи методу SADT.

Результат бізнес-процесу – те, заради чого здійснюється бізнес-процес, тобто діяльність завжди розглядається разом з метою цієї діяльності - отримання на виході деякого результату, що задовольняє заданим вимогам. Результати бізнес-процесу часто згадуються як виходи бізнес-процесу.

Власник бізнес-процесу – посадова особа, яка несе відповідальність за отримання результату процесу і володіє повноваженнями для розпорядження ресурсами, необхідними для виконання процесу.

Виконавці бізнес-процесу – команда фахівців з різних функціональних областей (крос-функціональна команда), що виконують дії процесу.

Входи бізнес-процесу – ресурси (матеріальні, інформаційні), необхідні для виконання і отримання результату процесу, які споживаються або перетворюються при виконанні процесу.

Управління процесу – як правило інформація, яка визначає правила перетворення входів в вихід.

Механізм процесу – то, що перетворює вхід у вихід. Механізмами, як правило, є співробітники (структурні підрозділи) організації та техніка, на якій вони працюють (верстати, оргтехніка).

В основі методології SADT лежать два основні принципи:

– SA-блоки;
– Декомпозиція.

На основі SA-блоків створюється ієрархічна багаторівнева модульна система, кожен рівень якої являє собою закінчену систему (блок), що підтримується та контролюється системою (блоком), що знаходиться над нею.

Використання декомпозиції дозволяє розділити кожен блок, який розуміється як єдине ціле, на свої складові, що описуються на більш детальній діаграмі. Процес декомпозиції проводиться до досягнення необхідного рівня подробиці опису. Діаграма обмежується 3-6 блоками для того, щоб деталізація здійснювалася поступово. Замість однієї громіздкої моделі використовується кілька невеликих взаємозалежних моделей, значення яких взаємно доповнюють одна одну, роблячи зрозумілою структуризацію складного об'єкта.

На рисунку 1.2 зображено розташування розглянутих елементів методу SADT при описі бізнес-процесів.

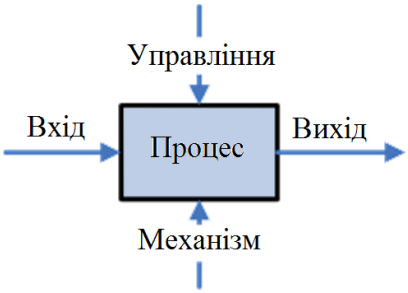


Рисунок 1.2 – Опис бізнес-процесу методом SADT

Розглянемо приклад схеми опису процесу закупівлі товару:
Вхід: гроші, потреба в товарі, інформація постачальника.
Вихід: товари, документи на товар.
Управління: правила тендерних закупівель, правила бухобліку (первинні документи).
Механізми: служба матеріально-технічного постачання, офісна техніка.
На рисунку 1.3 продемонстровано результат опису поставленої задачі.

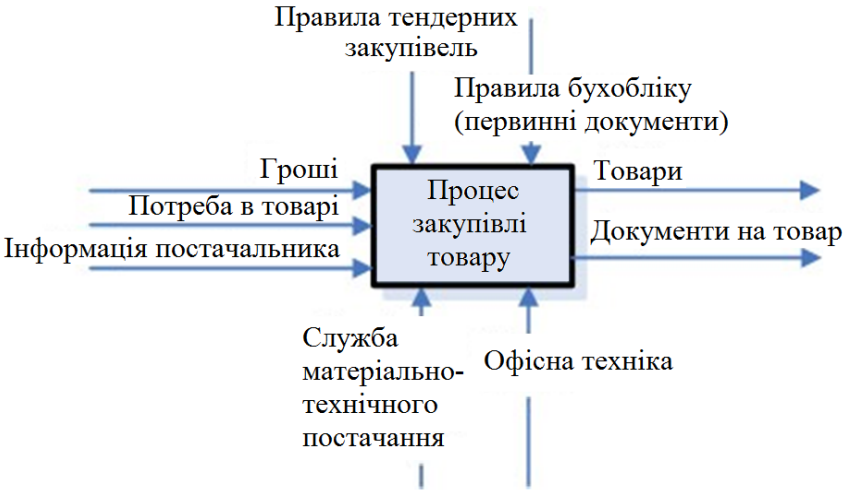


Рисунок 1.3 – Опис процесу закупівлі товару методом SADT

Розглянемо приклад, де вихід одного процесу може бути входом (управлінням, механізмом) для іншого (рисунок 1.4).

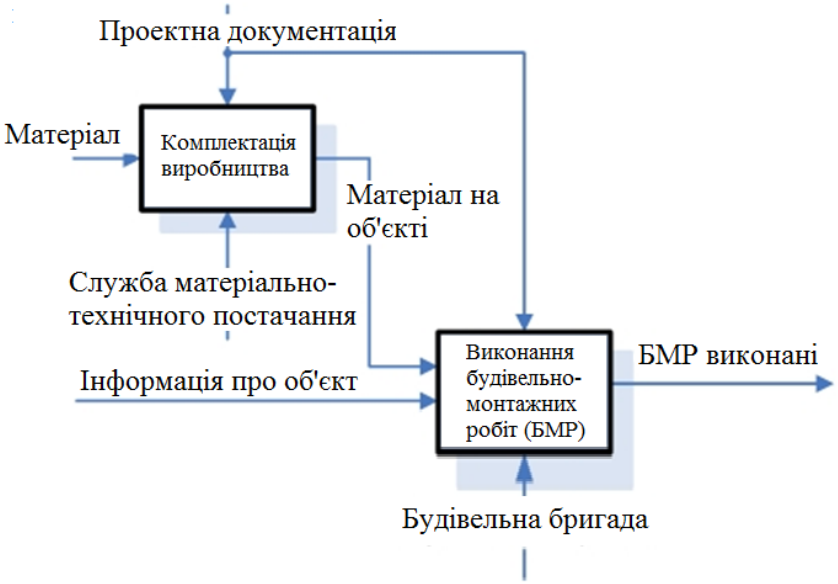


Рисунок 1.4 – Приклад взаємодії двох процесів

Запитання для самоперевірки

1. Що таке бізнес-процес?
2. Що таке моделювання бізнес-процесу?
3. Які види бізнес-процесів ви знаєте?
4. Що таке процесний підхід?
5. Які способи опису бізнес-процесів ви знаєте?
6. Коли який способи опису бізнес-процесів бажано використовувати?
7. З яких послідовних етапів складається процес моделювання в SADT?
8. Що таке результат бізнес-процесу?
9. Хто такий власник бізнес-процесу?
10. Хто такі виконавці бізнес-процесу?
11. Що таке входи бізнес-процесу?
12. Що таке управління процесу?

Лекція 2. Нотація IDEF0

1. Компоненти синтаксису IDEF0
2. Контекстна діаграма IDEF0
3. Приклади використання нотації IDEF0

1. Компоненти синтаксису IDEF0

IDEF0 – методологія функціонального моделювання. Використовується для створення функціональної моделі, що відображає структуру і функції системи, а також потоки інформації і матеріальних об'єктів, що зв'язують ці функції.

Для нових систем застосування IDEF0 направлено на визначення вимог і зазначення функцій для подальшої розробки системи, що відповідає поставленим вимогами і реалізує виділені функції.

Стосовно до вже існуючих систем IDEF0 може бути використана для аналізу функцій, виконуваних системою, і відображення механізмів, за допомогою яких ці функції виконуються.

Результатом застосування IDEF0 до деякої системи є модель цієї системи, що складається з ієрархічно упорядкованого набору діаграм, тексту документації та словників, пов'язаних один з одним за допомогою перехресних посилань.

Компоненти синтаксису IDEF0:

– блоки,
– стрілки,
– діаграми,
– правила.

Блоки представляють функції, які визначаються як діяльність, процес, операція, дія або перетворення. Блок описує функцію. Усередині кожного блоку міститься його ім'я і номер. Ім'я повинно бути активним дієсловом або дієслівним оборотом, що описує функцію. Номер блоку розміщується в правому нижньому кутку. Приклад позначення (рисунок 2.1):

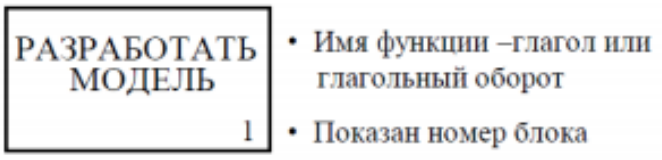


Рисунок 2.1 – Приклад позначення блоку

Стрілки являють дані або матеріальні об'єкти, пов'язані з функціями. Стрілки не уявляють потік або послідовність подій. Вони показують, які дані або матеріальні об'єкти мають надійти на вхід функції для того, щоб ця функція могла виконуватися.

Стрілки діляться на п'ять видів :

– входу (входять в ліву грань роботи) – зображують дані або об'єкти, що змінюються в ході виконання роботи;
– виходу (виходять з правої межі роботи) – зображують дані або об'єкти, що з'являються в результаті виконання роботи;
– управління (входять у верхню грань роботи) – зображують правила і обмеження, згідно з якими виконується робота;
– механізму (входять в нижню межу роботи) – зображують ресурси, необхідні для виконання роботи, але не змінюються в процесі роботи (наприклад, обладнання, людські ресурси тощо);

– виклику (виходять з нижньої межі роботи) – зображують зв'язку між різними діаграмами або моделями, вказуючи на деяку діаграму, де дана робота розглянута більш докладно.

Приклад позначення блоку та стрілок (рисунок 2.2):

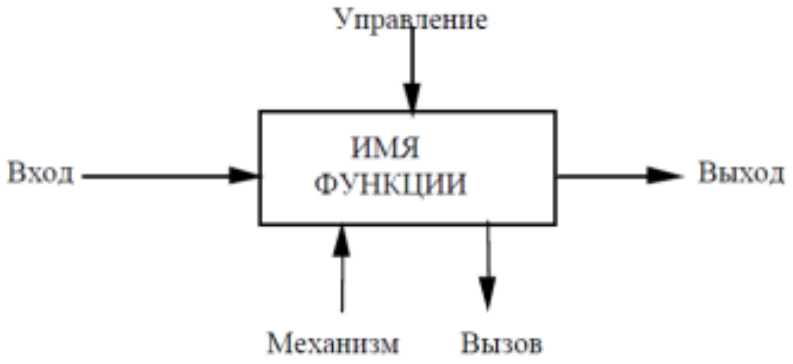


Рисунок 2.2 – Приклад позначення блоку та стрілок

Приклад діаграми IDEF0 для банківського завдання продемонстровано на рисунку 2.3.

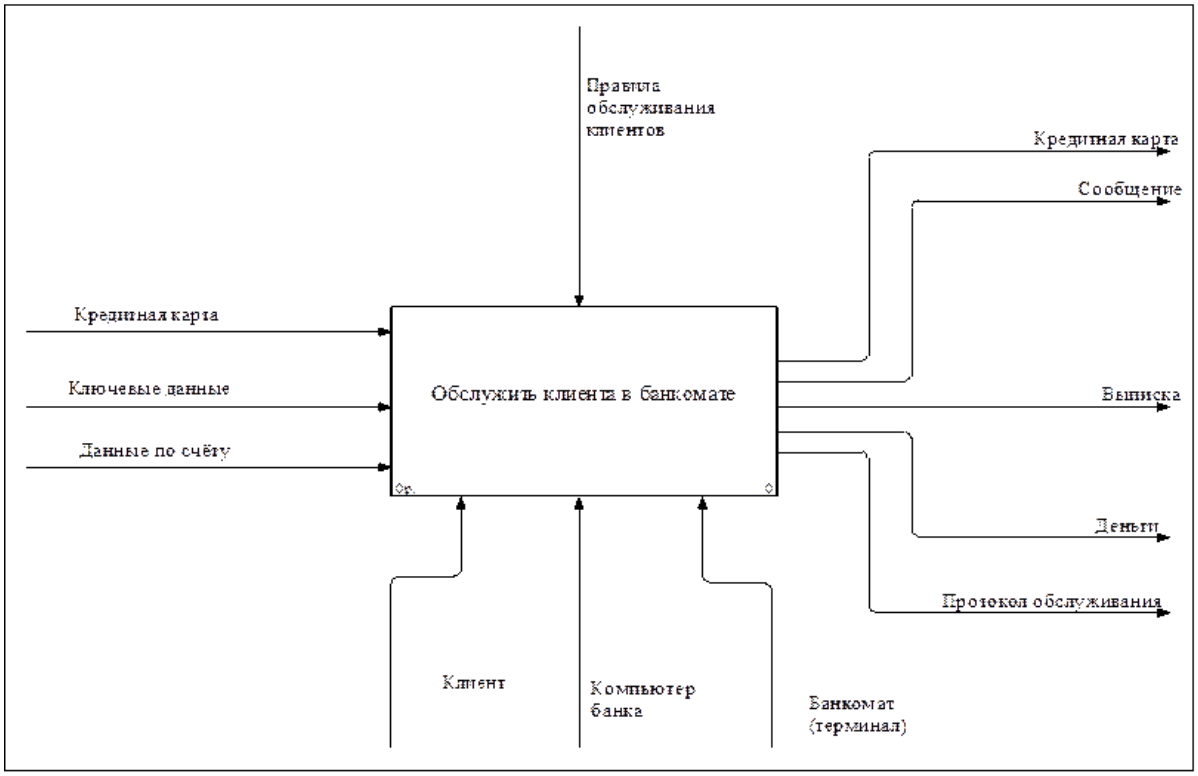


Рисунок 2.3 – Приклад процесу «Обслужити клієнта в банкоматі»

2. Контекстна діаграма IDEF0

IDEF0-моделі складаються з трьох типів документів: графічних діаграм, тексту і глосарію.

Перша діаграма в ієрархії діаграм IDEF0 завжди зображує функціонування системи в цілому. Такі діаграми називаються *контекстними* і позначаються А-0 (А мінус нуль).

У контекст входять:

– опис мети моделювання,

– області (опис того, що буде розглядатися як компонент системи, а що - як зовнішній вплив)
– точка зору (позиція, з якої буде будуватися модель).

Зазвичай в якості точки зору вибирається точка зору особи або об'єкта, відповідального за роботу системи, що моделюється в цілому (рисунок 2.4).

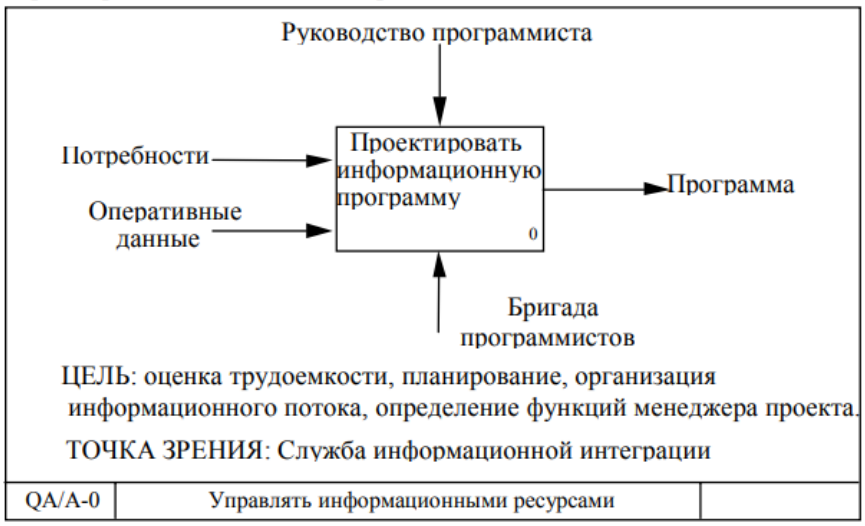


Рисунок 2.4 – Приклад діаграми A-0

IDEF0 модель має єдину мету і єдиний суб'єкт. *Мета моделі* – отримання відповідей на певну сукупність запитань. *Суб'єкт* – це сама система.

Методологія IDEF0 вимагає, щоб створювана модель системи розглядалася завжди з однієї і тієї ж позиції, або точки зору. Після визначення точки зору, з якої описується модель, створюється список даних, а потім список функцій.

Кожен блок діаграми IDEF0-моделі може бути деталізований на іншій діаграмі. Оскільки кожен блок розуміється як окремий, повністю певний об'єкт, поділ такого об'єкта на його структурні частини (блоки і дуги, складові діаграму) називається *декомпозицією*.

Декомпозиція формує кордони, і кожен блок в IDEF0 розглядається як формальна межа деякої частини описуваної системи (рисунок 2.5). Ця діаграма, звана *діаграмою-нащадком*, описує все, пов'язане з цим блоком і його дугами, і не описує нічого поза цією межею. Декомпозіруємий блок називається *батьківським блоком*, а що містить його діаграма – *батьківської діаграмою*.

Виділяють п'ять основних видів комбінованих стрілок:

– Вихід – вхід,
– Вихід – управління,
– Вихід – механізм виконання,
– Вихід – зворотний зв'язок на управління,
– Вихід – зворотний зв'язок на вхід.

Стрілка вихід – вхід застосовується, коли один з блоків повинен повністю завершити роботу перед початком роботи іншого блоку.



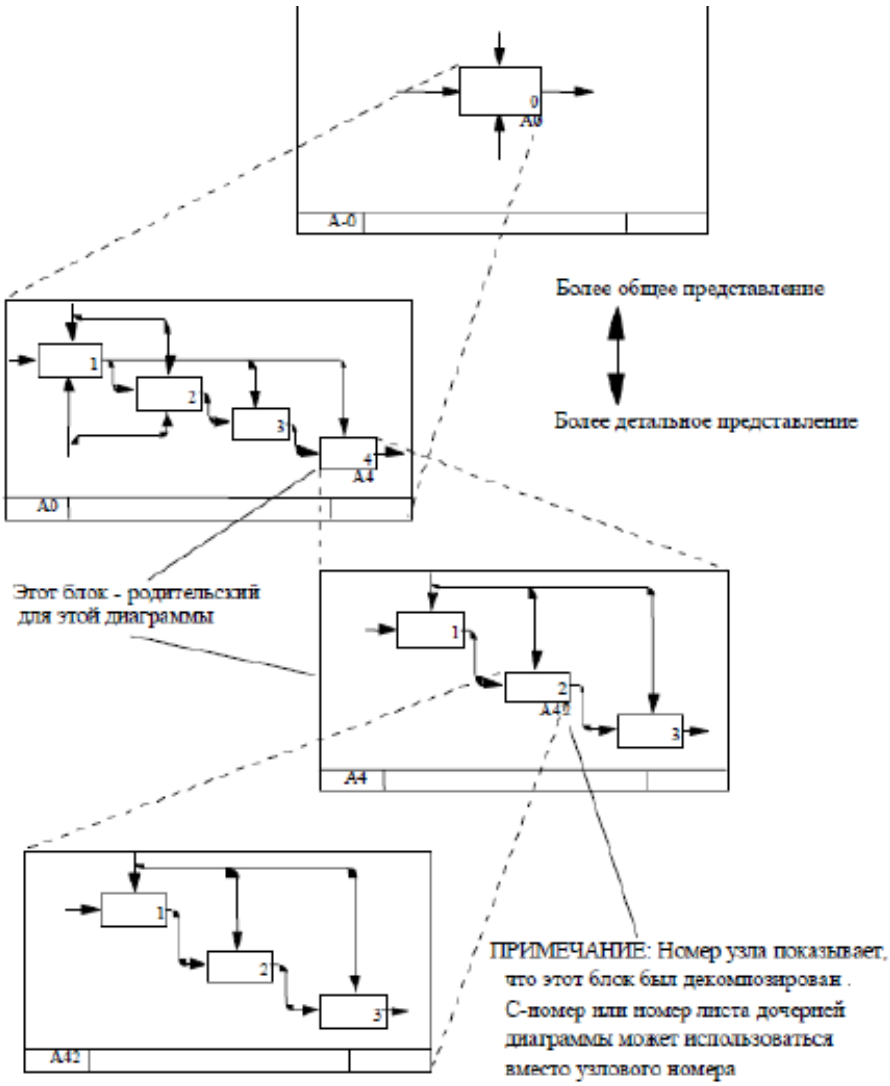
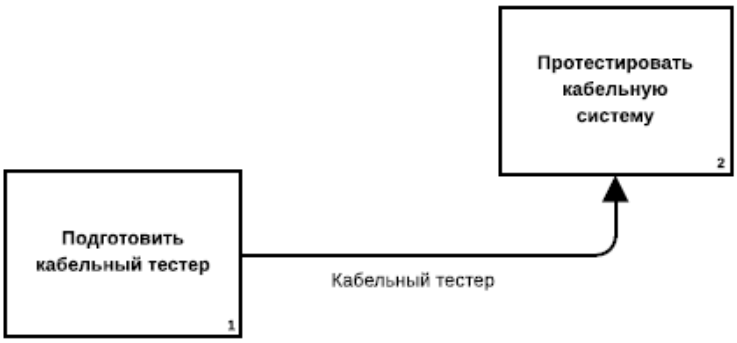


Рисунок 2.5 – Приклад декомпозиції

Стрілка вихід – управління показує, що один блок управляє роботою іншого.



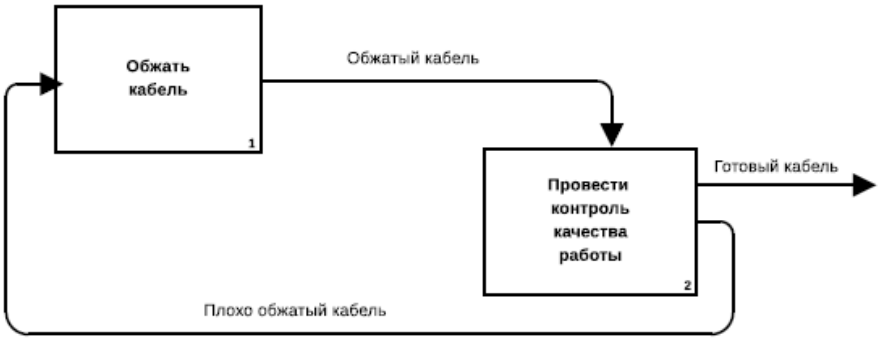
Стрілки вихід – механізм виконання показують, що вихід одного функціонального блоку застосовується в якості інструментарію для роботи другого.



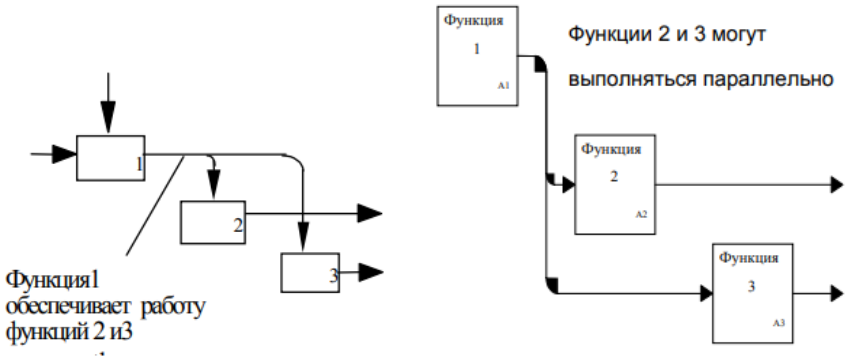
Стрілка вихід – зворотний зв'язок на управління застосовується в разі, коли залежний блок коригує виконання керуючого блоку.



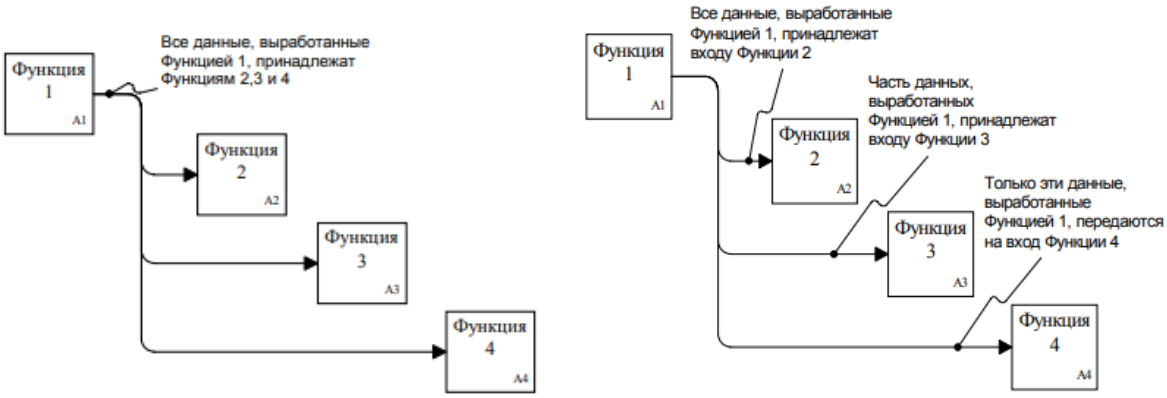
Стрілка вихід – зворотний зв'язок на вхід зазвичай застосовується для опису циклів повторної обробки чогось. Стрілка зображується під блоком. Крім того, можливе застосування даного зв'язку при повторному використанні бракованої продукції.



Різні функції в моделі можуть бути виконані паралельно, якщо задовольняються необхідні обмеження (умови). Приклад позначення:



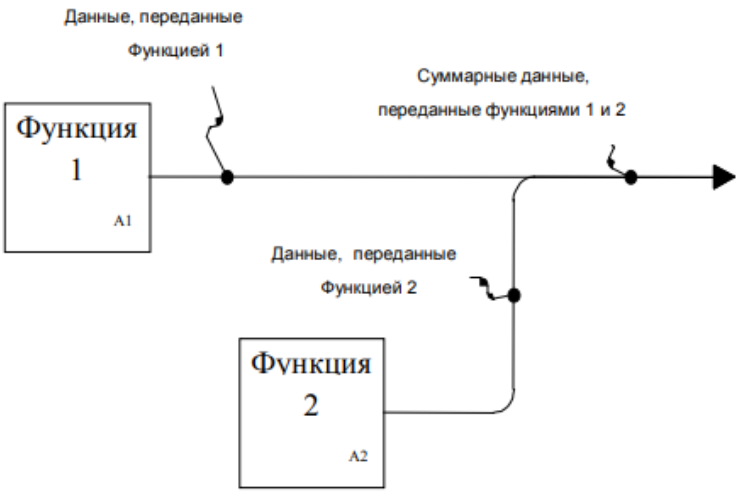
Розгалуження і злиття стрілок покликане зменшити завантаженість діаграм графічними елементами (лініями). Щоб стрілки і їх сегменти правильно описували зв'язку між блоками – джерелами і блоками – споживачами, використовується апарат міток. Мітки зв'язуються з сегментами за допомогою тильд.



Роз'єднані або об'єднані стрілки можуть мати найменування, що відрізняються від найменування вихідної стрілки. Сукупність вихідної і роз'єднаних або об'єднаних стрілок називається *пов'язаними стрілками*. Ця техніка застосовується для того, щоб відобразити використання тільки частини сировини або інформації, які охоплюють вихідної стрілкою.



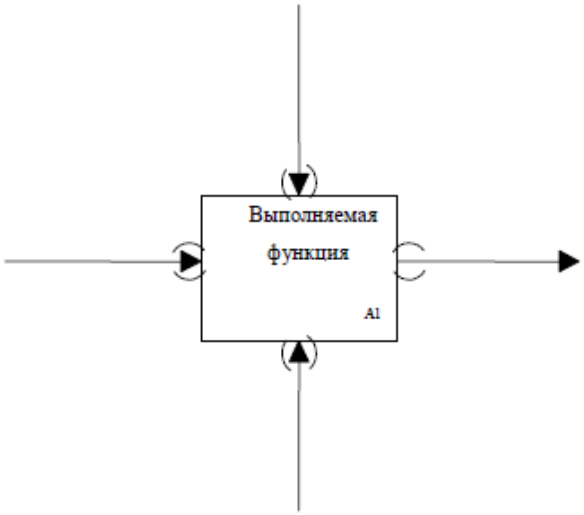




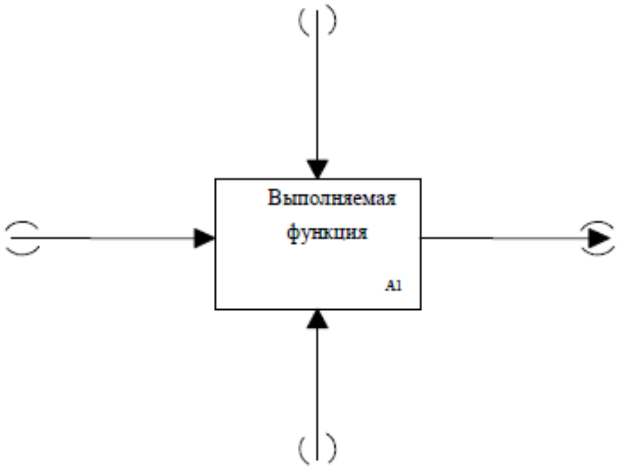
Тунель – круглі дужки на початку і / або закінчення стрілки. Тунельні стрілки означають, що дані, виражені цими стрілками, не розглядаються на батьківській діаграмі і / або на дочірній діаграмі (зазвичай у зв'язку з несуттєвими на певному рівні абстракції).

Стрілка, що виходить з тунелю, називається *стрілкою імпорту ресурсів*. Стрілка, що входить в тунель, називається стрілкою *подразумевання ресурсу*.

Стрілка, вміщена в тунель там, де вона приєднується до блоку означає, що дані, виражені цієї стрілкою, не обов'язкові на наступному рівні декомпозиції.



Стрілка, що поміщається в тунель на вільному кінці означає, що виражені нею дані відсутні на батьківській діаграмі.



Приклад тунелей (рисунок 2.7).

Правила побудови діаграм:

– У складі моделі має бути присутня контекстна діаграма А-0, яка містить тільки один блок. Номер єдиного блоку на контекстній діаграмі А-0 повинен бути 0.
– Не контекстні діаграми повинні містити не менше трьох і не більше шести блоків.
– Блоки на діаграмі повинні розташовуватися по діагоналі - від лівого верхнього кута діаграми до правого нижнього в порядку присвоєних номерів. Блоки на діаграмі, розташовані вгорі ліворуч домінують над блоками, розташованими внизу праворуч.
– Імена блоків і мітки стрілок повинні бути унікальними. Якщо мітки стрілок збігаються, це означає, що стрілки відображають ідентичні дані.
– Блоки завжди повинні мати хоча б одну керуючу і одну вихідну стрілку, але можуть не мати вхідних стрілок.
– Якщо одні й ті ж дані служать і для управління, і для входу, малюється тільки стрілка управління. Цим підкреслюється керуючий характер даних і зменшується складність діаграми.

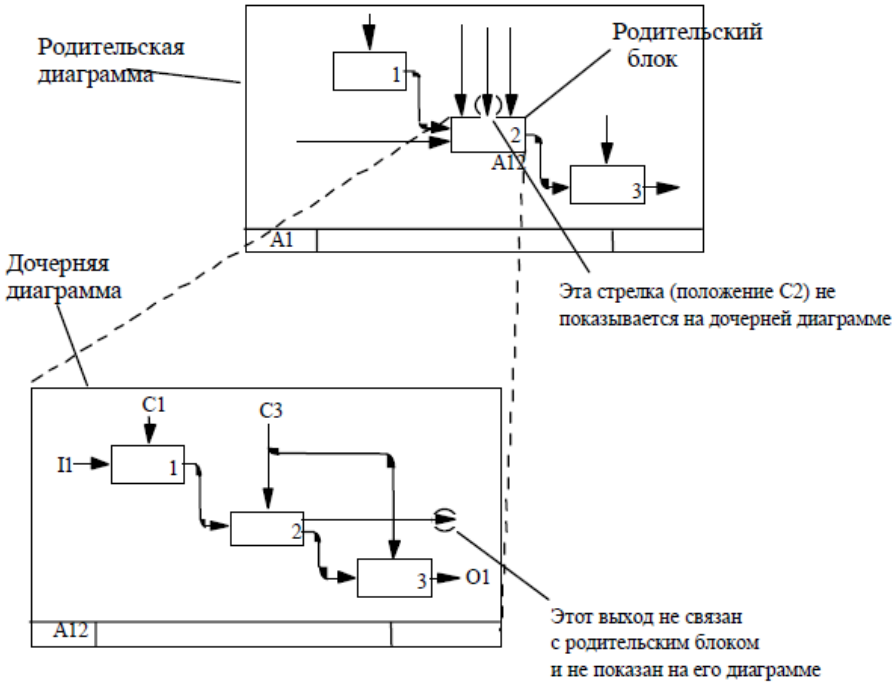


Рисунок 2.7 – Приклад тунелей

3. Приклади використання нотації IDEF0

Приклад контекстних діаграм процесу «Написання статті» продемонстровані на рисунку 2.8 та 2.9.

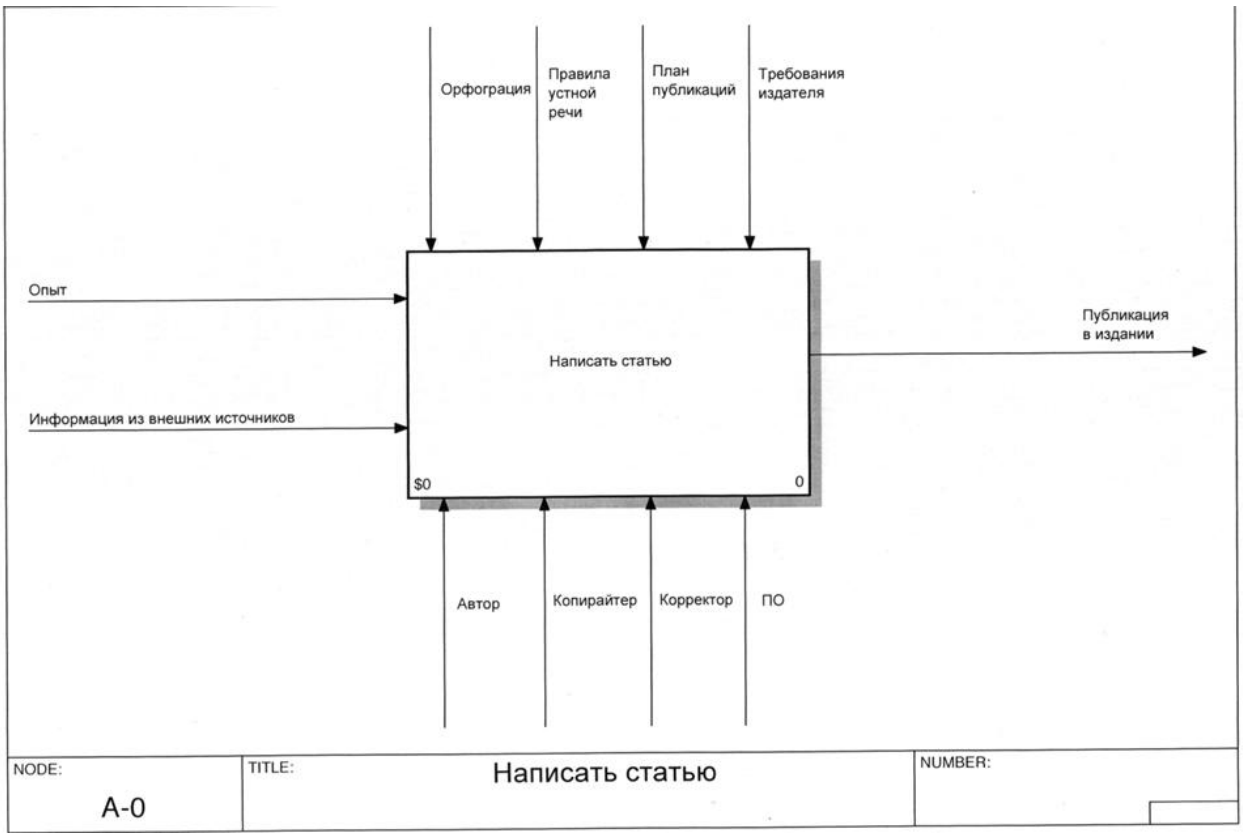


Рисунок 2.8 – Приклад контекстної діаграми А-0 процесу «Написання статті»

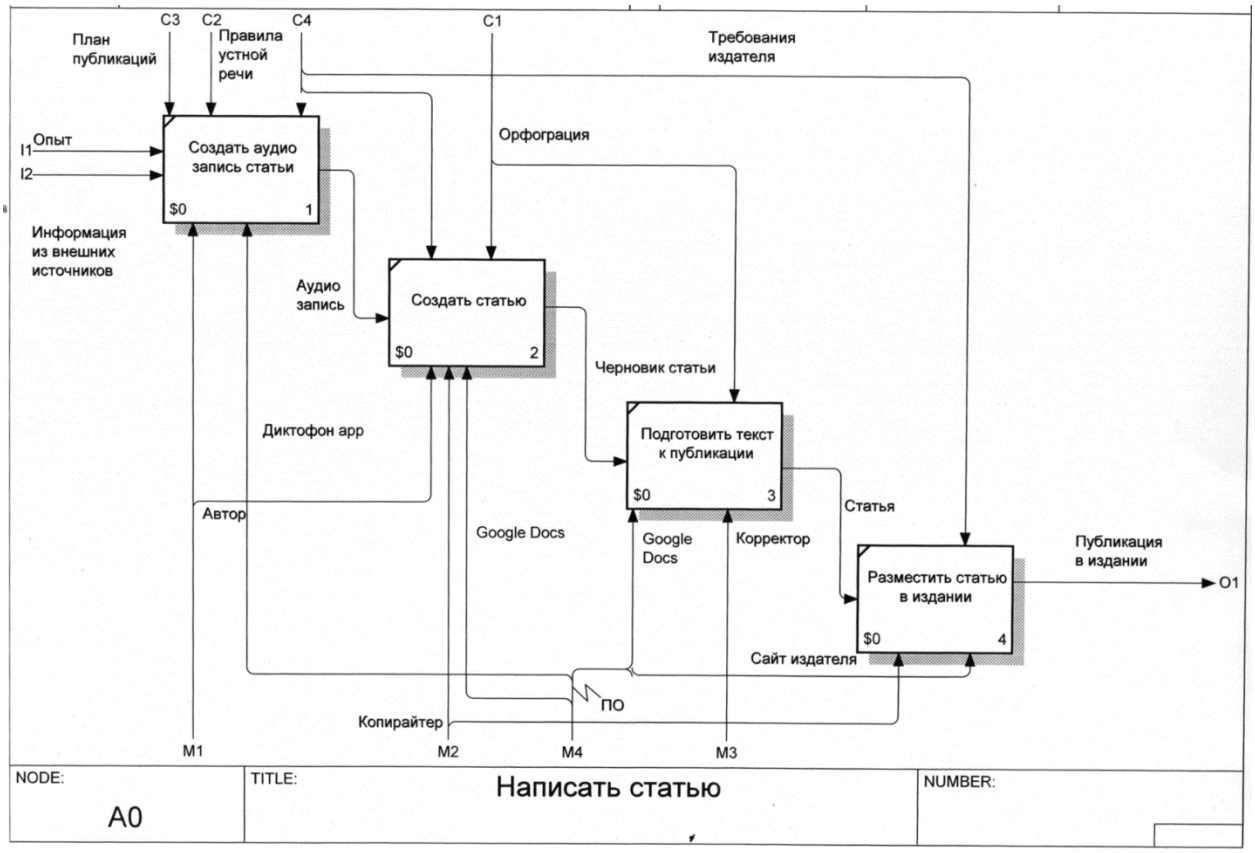


Рисунок 2.9 – Приклад контекстної діаграми А0 процесу «Написання статті»

Приклад контекстних діаграм процесу «Приготування борщу» продемонстровані на рисунку 2.10 та 2.11.



Рисунок 2.10 – Приклад контекстної діаграми А-0 процесу «Приготування борщу»

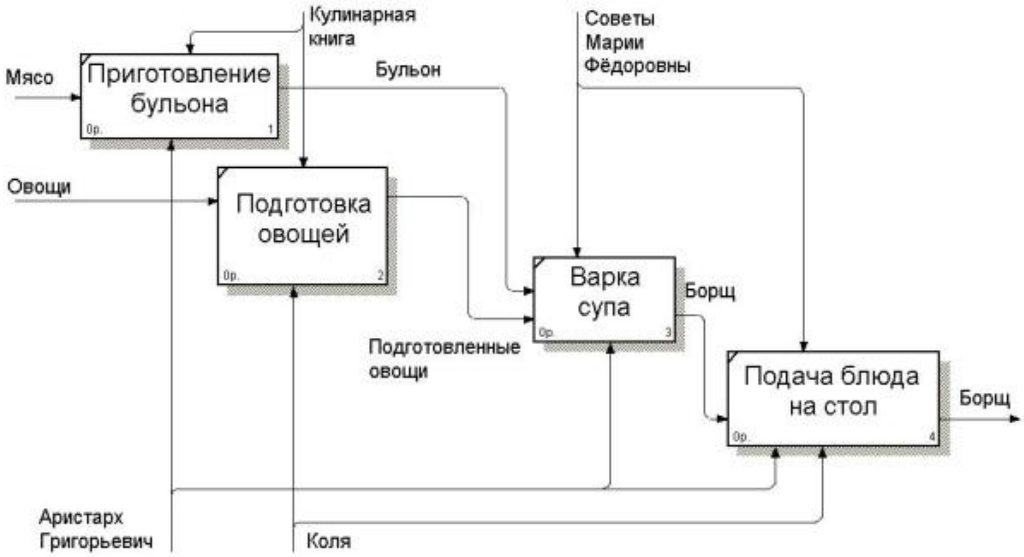


Рисунок 2.11 – Приклад контекстної діаграми А-0 процесу «Приготування борщу»

Для створення моделі розглянемо діяльність компанії, яка займається виготовленням та продажом напівфабрикатів – швидкозаморожених пельменів і вареників.

Послідовність основних процедур в компанії така:

– менеджери приймають замовлення від клієнтів;
– робочі виготовляють продукцію;
– робочі проводять контроль якості продукції;
– комірник відвантажує клієнтам замовлення

Блоки діаграми декомпозиції А0

Ім’я	Опис
1. Продаж та маркетинг	Маркетингові дослідження
2. Виготовлення та контроль якості	Виготовлення та контроль якості пельменів та вареників
3. Відвантаження та отримання	Відвантаження замовлень клієнтам та отримання сировини від постачальників

Блоки діаграми декомпозиції А2

Ім’я	Опис
Управління виготовленням та контролем якості	Перегляд замовлень, перегляд результатів контролю якості, формування груп

	замовлень на виготовлення та відвантаження
Виготовлення пельменів	Виготовлення пельменів відповідно до рецептури та вказівок майстра
Виготовлення вареників	Виготовлення вареників відповідно до рецептури та вказівок майстра
Контроль якості	Контроль якості готової продукції. Відправлення браку на переробку

Приклад контекстних діаграм процесу «Виготовленням та продаж напівфабрикатів» продемонстровані на рисунках 2.12 – 2.14.



Рисунок 2.12 – Приклад контекстної діаграми А-0 процесу «Виготовленням та продаж напівфабрикатів»

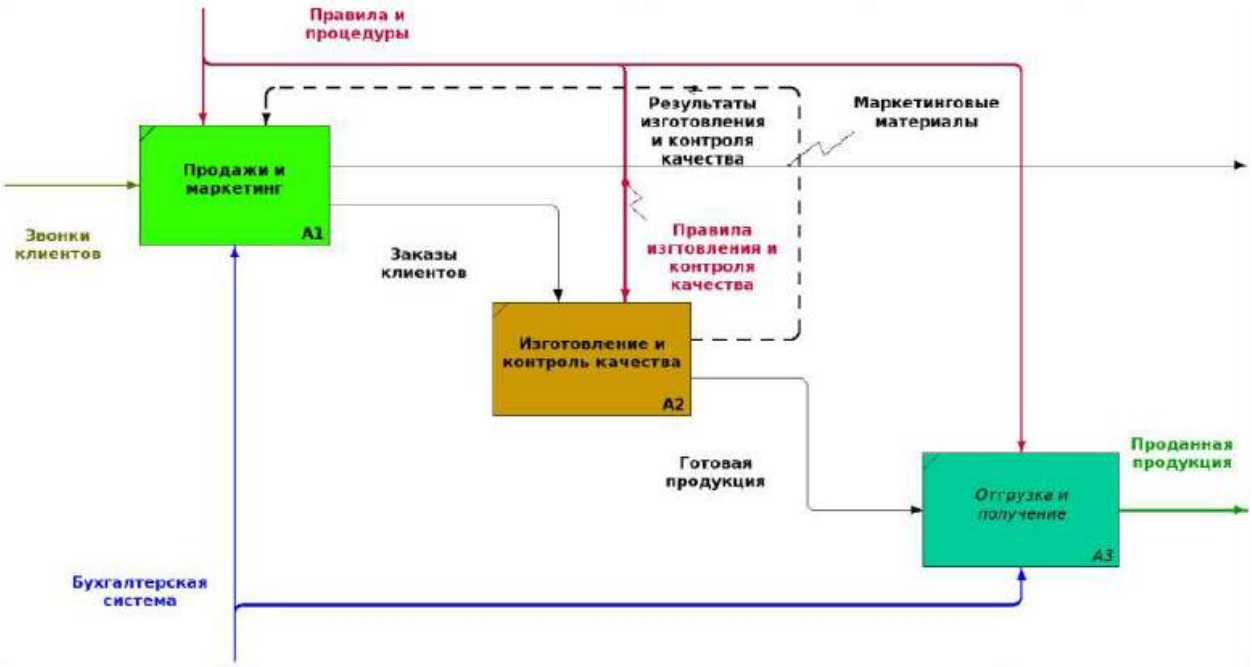


Рисунок 2.13 – Приклад контекстної діаграми А0 процесу «Виготовленням та продаж напівфабрикатів»

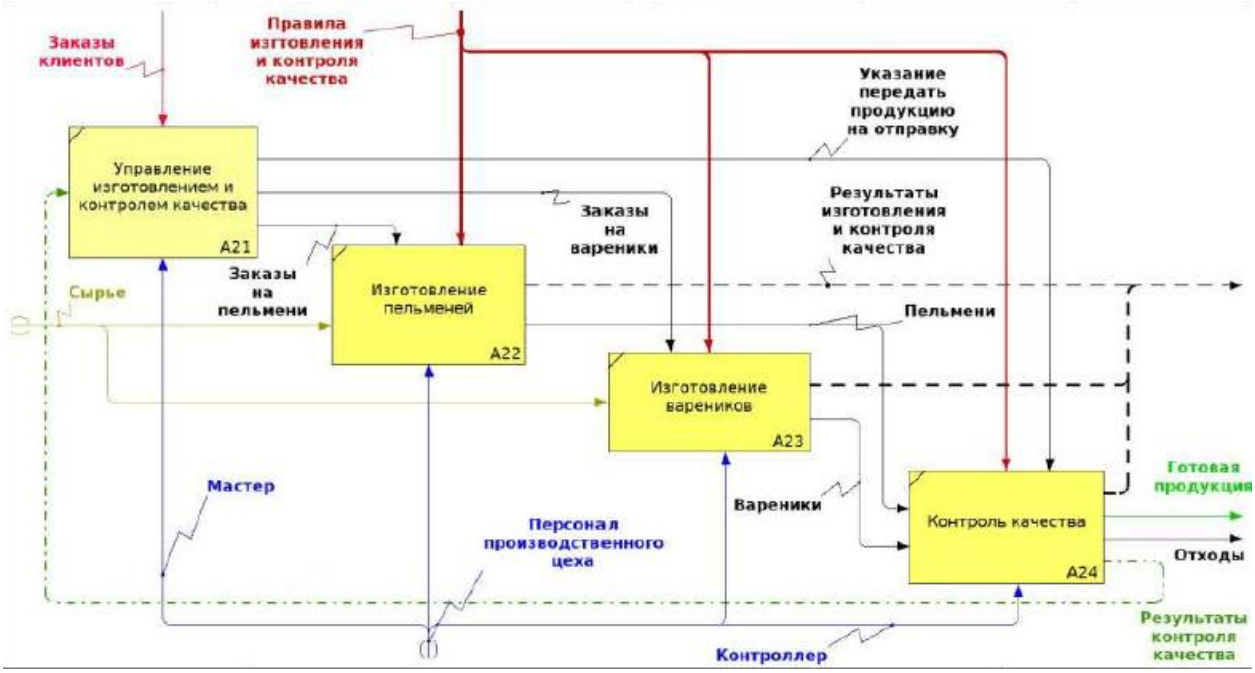


Рисунок 2.14 – Приклад контекстної діаграми А2 процесу «Виготовленням та продаж напівфабрикатів»

Запитання для самоперевірки

1. З яких елементів складається нотація IDEF0?
2. Наведіть правила розташування стрілок.
3. Що таке тунель?
4. Наведіть контекстні діаграми А-0 та А0 для процесів «Реєстрація користувача на сайті» та «Захист інформації від спотворення»?

Лекція 3. Нотація IDEF3

1. Компоненти синтаксису IDEF3
2. Діаграма PFDD

1. Компоненти синтаксису IDEF3

Стандарт IDEF3 – це методологія опису процесів, що розглядає послідовність виконання і причинно-наслідкові зв'язки між ситуаціями і подіями для структурного представлення знань про систему.

За допомогою IDEF3 описують логіку виконання робіт, черговість їх запуску і завершення, тобто IDEF3 надає інструмент моделювання сценаріїв дій співробітників організації, відділів, цехів тощо, наприклад, порядок обробки замовлення або події, на які необхідно реагувати за кінцевий час, виконання дій з виробництва товару тощо.

Дана методика не має жорстких синтаксичних і семантичних обмежень.

Дуже часто IDEF3 використовують як метод, що доповнює IDEF0. Кожен функціональний блок IDEF0 може бути представлений у вигляді окремого процесу IDEF3.

IDEF3 як інструмент моделювання фіксує наступну інформацію про процес:

- об'єкти, які беруть участь при виконанні сценарію;
- ролі, які виконують ці об'єкти, наприклад, агент, транспорт;
- відносини між роботами в ході виконання сценарію процесу;
- стан і зміни, яким піддаються об'єкти;
- час виконання і контрольні точки синхронізації робіт;
- ресурси, які необхідні для виконання робіт.

Засоби документування та моделювання IDEF3 дозволяють виконувати наступні завдання:

- документувати наявні дані про технології виконання процесу, виявлені, наприклад, під час опитування фахівців предметної області, відповідальних за організацію даного процесу або беруть участь в ньому;
- аналізувати існуючі процеси і розробляти нові;
- визначати і аналізувати точки впливу потоків супутнього документообігу на сценарій технологічних процесів;
- визначати ситуації, в яких потрібно прийняття рішення, що впливає на життєвий цикл процесу, наприклад, зміна конструктивних, технологічних або експлуатаційних властивостей кінцевого продукту;
- сприяти прийняттю оптимальних рішень при реорганізації процесів;
- розробляти імітаційні моделі технологічних процесів за принципом «як буде, якщо ...».

У IDEF3 використовується два типи діаграм, що представляють опис одного і того ж сценарію в різних ракурсах.

- Діаграм опису послідовності етапів процесу (Process Flow Description Diagrams, PFDD);
- Діаграми переходу стану об'єкта (Object State Transition Network, OSTN).

За допомогою *діаграм опису послідовності етапів процесу* (PFDD) документується послідовність і опис стадій обробки в рамках досліджуваного бізнес-процесу. Опис проводиться з точки зору стороннього спостерігача. Ключовими елементами є поняття, процес, логіка процесу.

Діаграми переходу стану об'єкта (OSTN) використовуються для ілюстрації трансформацій, які відбуваються на кожній стадії бізнес-процесу. При цьому опис проводиться з точки зору самого об'єкта.

2. Діаграма PFDD

Розглянемо елементи діаграми опису послідовності етапів процесу (PFDD). Основними елементами IDEF3-моделі є (рисунок 3.1):

– одиниці робіт;
– зв'язки;
– перехрестя;
– об'єкти посилань.

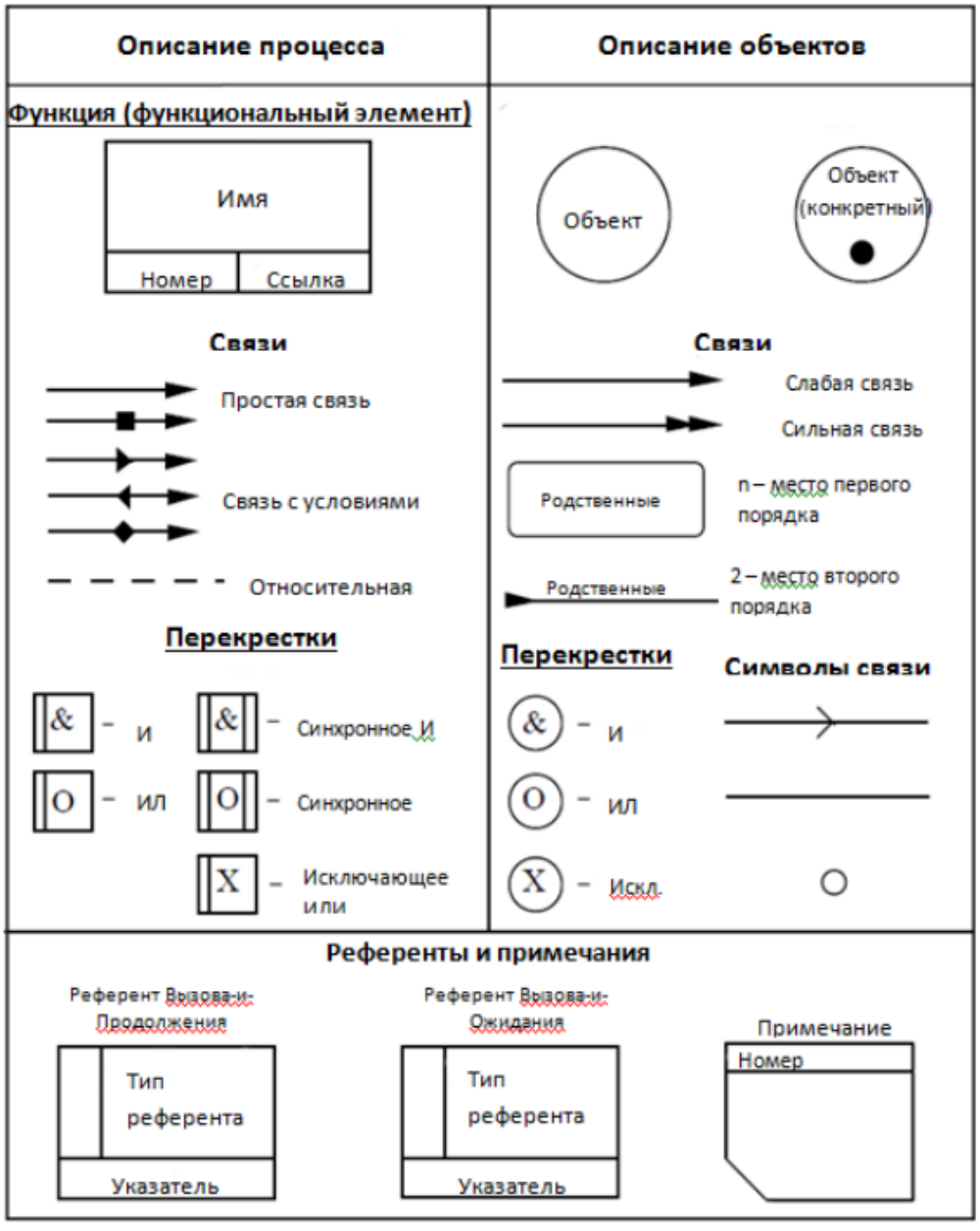
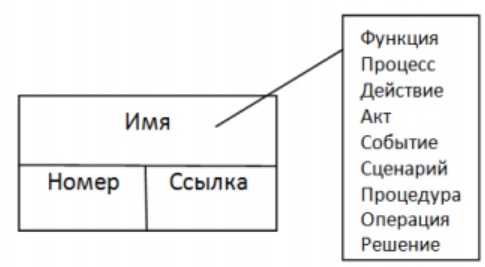


Рисунок 3.1 – Позначення основних елементів IDEF3-моделі

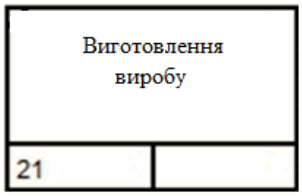
Опис процесу являє всілякі ситуації (процеси, функції, дії, акти, події, сценарії, процедури, операції або рішення), які можуть відбуватися в системі, що моделюється в логічних та тимчасових відносинах.

Дія в IDEF3 називається *одиницею роботи* (Unit of Work, *UOW*) і позначається прямокутником. Дії іменуються дієсловами або віддієслівним іменниками. Кожній дії призначається унікальний номер. Номер ідентифікатора процесу призначається послідовно.

Позначення дії:



Приклад застосування:



У правому нижньому кутку UOB-елемента розташовується посилання і використовується для вказівки посилань або на елементи з функціональної моделі IDEF0, або для вказівки на відділи або конкретних виконавців, які будуть виконувати зазначену роботу.

Елемент зв'язку необхідний для організації відносин між елементами діаграми і опису динаміки процесів, що відбуваються. Зв'язки використовуються насамперед для позначення відносин між процесами, відображення часовій послідовності виконання сценаріїв в діаграмах опису процесу.

Усі зв'язки односпрямовані. Зазвичай стрілки малюють зліва направо, що виходять з правої і входять в ліву сторону блоків, або зверху вниз.

Зв'язки між функціональними блоками можуть бути:

– тимчасові,
– логічні,
– причинно-наслідкові,
– природні,
– звичайні.

У переважній більшості випадків використовуються зв'язки, що відображають просте тимчасове відношення між блоками.

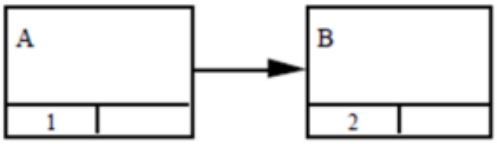
Існує два основних типи зв'язків, які використовуються в IDEF3 схемах:

– зв'язки пріоритету (старшинства)
– відносні (переривчасті) зв'язки.

Зв'язки простої черговості демонструють тимчасовій пріоритет відносин між функціональними блоками UOB. Вони є найбільш широко використовуваними зв'язками і позначаються суцільною стрілкою, а іноді додатковим маркером, прикріпленим до стовбура стрілки.



Приклад позначення простої черговості:

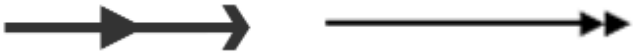


Приклад застосування:



Існують наступні підвиди зв'язку простої черговості:

Зв'язок *Об'єктний потік* застосовується в разі, коли об'єкт, що є результатом виконання вихідного дії, необхідний для виконання кінцевого дії.



Зв'язок виглядає, як лінія з двома стрілками, але розташована посередині спрямована в протилежний бік. Цей варіант обмеження вказує на те, що елемент В (другий по послідовності) може бути виконаний раніше елемента А (перший по порядку). Більш того, для того, щоб дії в елементі А були успішно завершені і можна було перейти до наступних етапів, елемент В повинен виконуватися в обов'язковому порядку.



Зв'язок виглядає, як лінія зі стрілкою на кінці та багатокутником («зірочкою») посередині. Означає, що елементи А і В взаємопов'язані, тобто виконання одного без іншого неможливо та безглуздо, причому, це правило працює в обидві сторони.

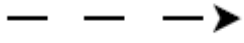


Зв'язок виглядає, як лінія зі стрілкою на кінці і квадратом посередині. Тут навпаки, послідовність дій може бути будь-якою, і взаємозв'язок між елементами мінімальний. Він буде залежати від рішення адміністратора та поточної ситуації.



Відносні (переривчасті) зв'язки не несуть ніякої певної семантики. З цієї причини їх часто називають користувацькими зв'язками (стрілка відносини). Цей тип посилань підкреслює існування (можливо, що обмежують) відносини між двома UOB.

Стрілка відносини (Relational Link) – пунктирна лінія, що використовується для зображення зв'язків між одиницями робіт (UOW), а також між одиницями робіт і об'єктами посилань. Значення задається аналітиком окремо для кожного випадку.



Усі зв'язки, які використовуються в схемі, в обов'язковому порядку маркуються і нумеруються.

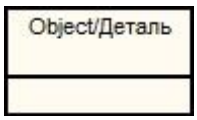
Для маркування використовуються абревіатури від назв типів зв'язків :

– СП або PL (від precedence links) – позначення для зв'язків передування;

– СО або DL (від dashed links) – зв'язку відносин.

Об'єкт посилання в IDEF3 виражає якусь ідею, концепцію або дані, які не можна пов'язати зі стрілкою, перехрестям або роботою. Вони використовуються в моделі для залучення уваги читача до якихось важливих аспектах моделі. При внесенні об'єктів посилань крім імені слід вказувати тип об'єкта посилання.

Приклад позначення:



Приклади використання:



Коли мова йде про процес, зазвичай припускають лінійну послідовність дій. На вході у нас якісь дані і поставлена задача, на виході після виконання певної послідовності дій – рішення. Але що робити, якщо на певному етапі процесу дії залежать від того, чи виконано певну умову?

Наприклад, працівник складу перевіряє документ на наявність підпису відповідальної особи. Якщо підпис присутній, товар видається. Якщо немає, документ повертається власнику і людина відправляється за потрібним підписом.

Для реалізації варіантів дій в залежності від виконання певних умов і були створені Вузли.

За кількістю вхідних і вихідних стрілок вузли в IDEF3 діляться на два види:

– вузли сходження, в яких сходяться гілки різних підпроцесів.

– вузли розбіжності, які ділять один процес на кілька «гілок».



Паралельний вузол має позначення – логічне І. Він вказує, що підпроцеси, які запускаються після вузла або, навпаки, були запущені перед вузлом, виконуються одночасно. Позначаються такі вузли символом & (логічне «І»). Все, що виходить з вузла з цим символом, запускається паралельно.

Наприклад, у нас є процес А, далі йде паралельний вузол, з якого виходять стрілки до процесів В, С і D. Інформація з процесу А відправляється в вузол, після чого запускаються всі вихідні процеси, тобто ті самі В, С і D.

За часом ці процеси можуть запускатися в довільному порядку. У деяких випадках процес В вже буде завершено, а процес D тільки почнеться. Ці нюанси зазвичай описуються додатково – текстом і за допомогою спеціальних діаграм.

Всі процеси після паралельного вузла обов'язково будуть запущені. І виконані вони будуть паралельно, тобто незалежно один від одного, кожен з них заснований тільки на результатах процесу А.

В позначенні альтернативного вузла присутня буква «О» або «Х».

– OR. У цьому випадку після вузла можуть запускатися один або кілька підпроцесів, в залежності від умови, що виконується в вузлі. Тобто гілки, які задовольняють умові, виконуються, інші – ні.
– XOR. У такому типі вузлів умова більш жорстка, в результаті виконується тільки одна з гілок, яка повністю відповідає умові.

Асинхронні вузли. Якщо процеси можуть запускатися асинхронно, тобто в різний час, то вузол має одну рису всередині прямокутника. При цьому, наприклад, процес В може бути запущений о 8 годині ранку, а процес С – о 2 годині дня або пізніше, в залежності від якихось умов. Позначається:



Синхронні вузли. Якщо нам важливо, щоб процеси після вузла були запущені одночасно, необхідно використовувати синхронний вузол. Він позначається двома вертикальними лініями всередині прямокутника – зліва і справа.



Синхронні вузли можуть бути не тільки вихідними, але і вхідними. В цьому випадку вузол активується тільки тоді, коли всі паралельні процеси закінчяться. Більш того, дуже важливо, щоб вони закінчилися одночасно.

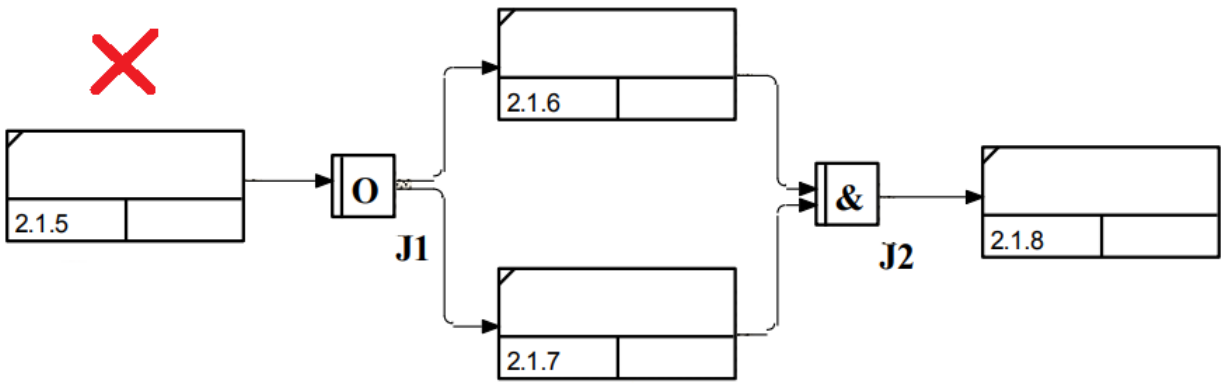
Повна синхронізація можлива тільки в разі автоматичних операцій. У бізнес-процесах беруть участь люди, тому хтось може закінчити роботу раніше, а хтось кілька затягне процес. Але в разі синхронного вузла, передати в нього результати роботи можна буде тільки одночасно, коли всі паралельні процеси будуть завершені.

Всі з'єднання на діаграмі повинні бути парними. Однак при цьому типи з'єднань не зобов'язані збігатися. На діаграмі з'єднання зазвичай позначаються літерою «J» та цифрою.

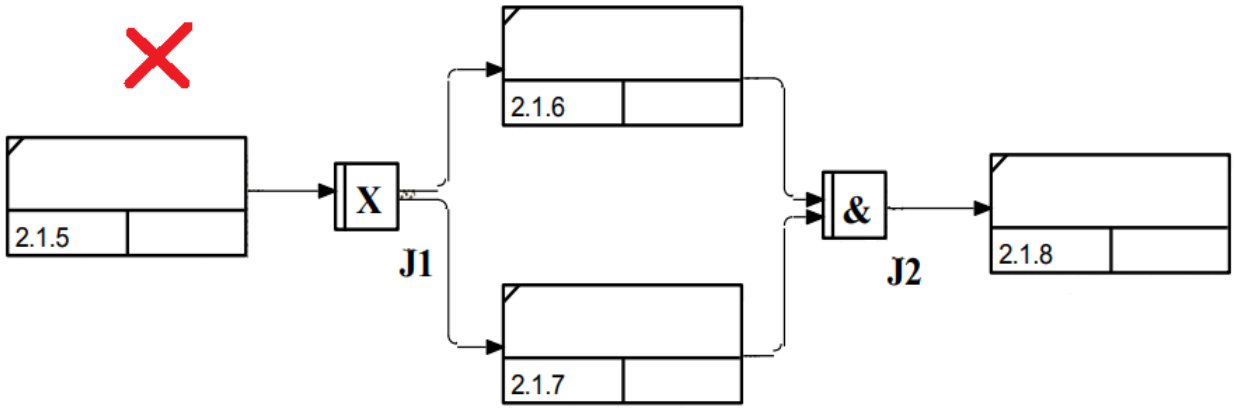
Графический символ	**Название элемента**	**Смысл в случае слияния стрелок** (Fan-in Junction)	**Смысл в случае разветвления стрелок** (Fan-out Junction)
&	Асинхронное «И» (Asynchronous AND)	Все предшествующие процессы должны быть завершены.	Все следующие процессы должны быть запущены.
&	Синхронное «И» (Synchronous AND)	Все предшествующие процессы завершаются одновременно.	Все следующие процессы запускаются одновременно.
O	Асинхронное «ИЛИ» (Asynchronous OR)	Один или несколько предшествующих процессов должны быть завершены.	Один или несколько следующих процессов должны быть запущены.
O	Синхронное «ИЛИ» (Synchronous OR)	Один или несколько предшествующих процессов завершаются одновременно.	Один или несколько следующих процессов запускаются одновременно.
X	Исключающее «ИЛИ» (XOR — Exclusive OR)	Только один предшествующий процесс завершён.	Только один следующий процесс запускается.

Правила створення перехресть :

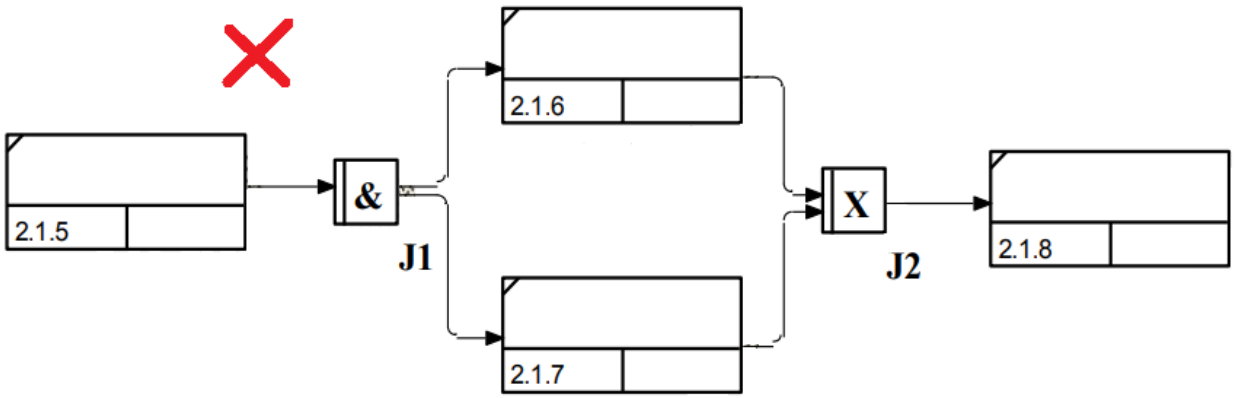
– Кожному перехрестю для злиття повинен передувати перехрестя для розгалуження.
– Перехрестя для злиття «І» не може слідувати за перехрестям для розгалуження типу синхронного або асинхронного «АБО».



– Перехрестя для злиття «І» не може слідувати за перехрестям типу виняткового «АБО».

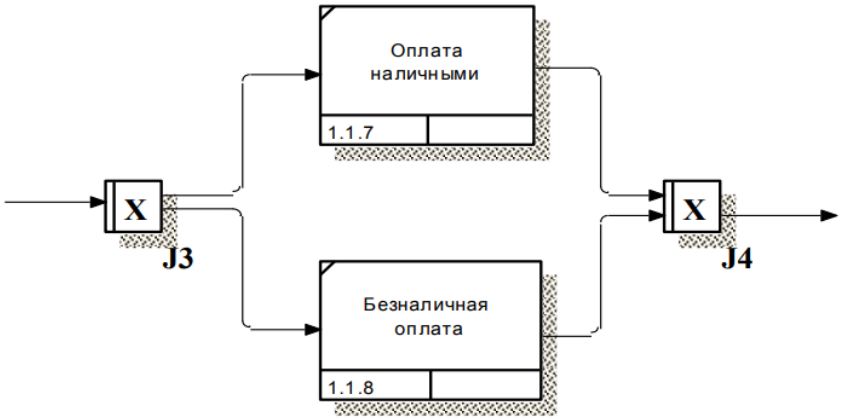


– Перехрестя для злиття типу виняткового «АБО» не може слідувати за перехрестям для розгалуження типу «І»

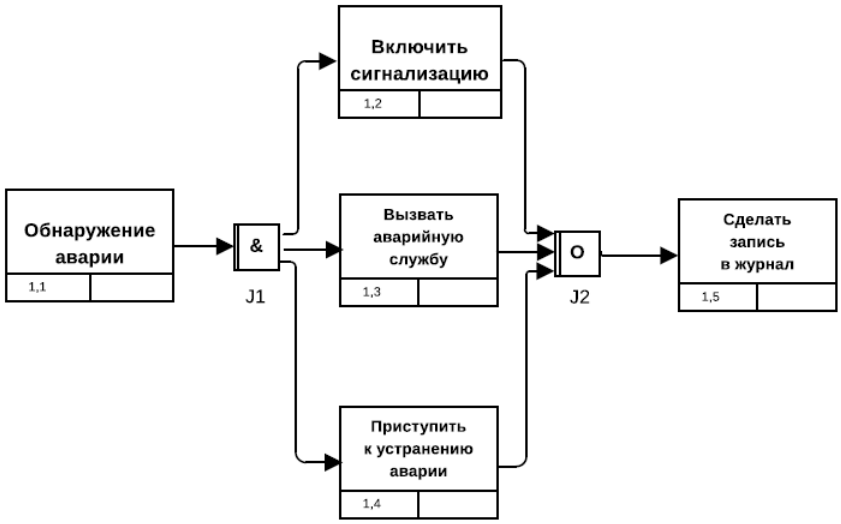


– Перехрестя, що має одну стрілку на одній стороні, повинен мати більше однієї стрілки на інший.

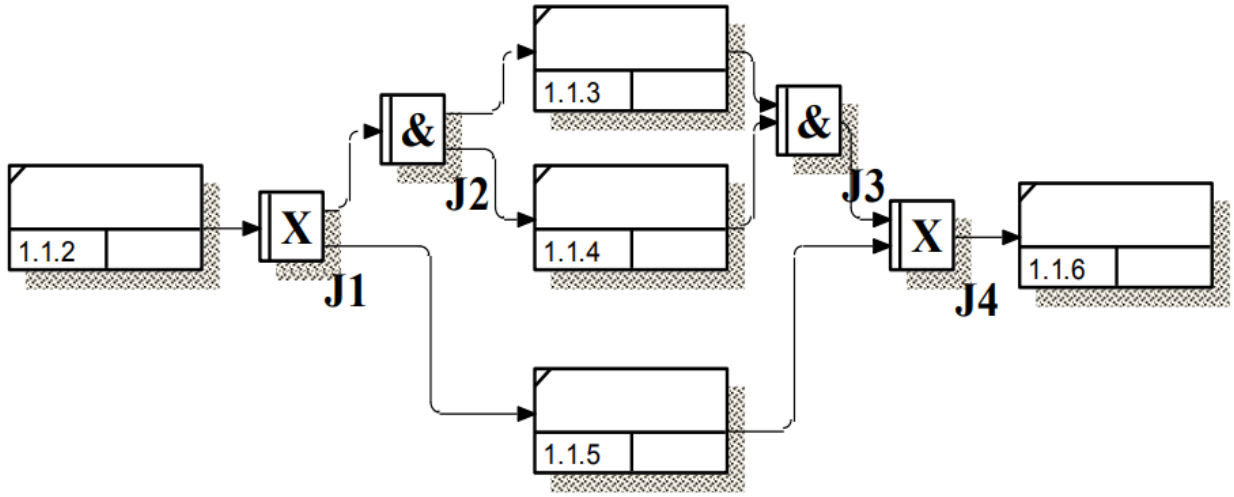
Розглянемо приклад 1:



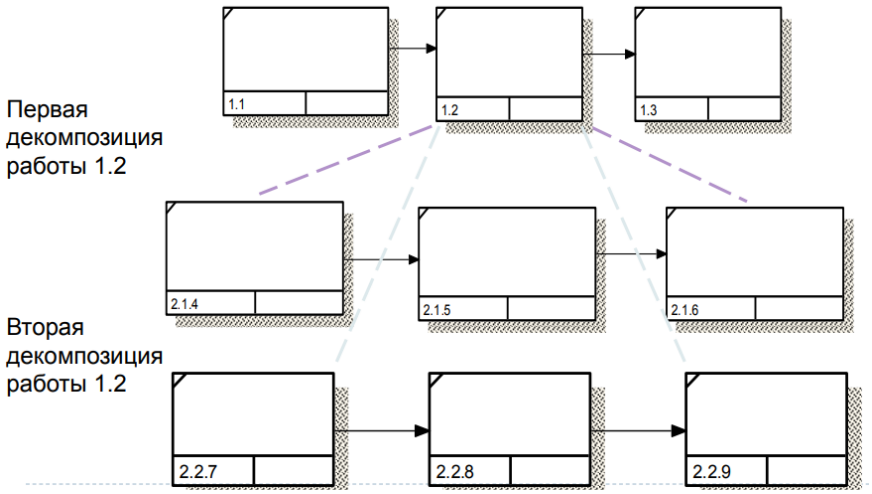
Розглянемо приклад 2:



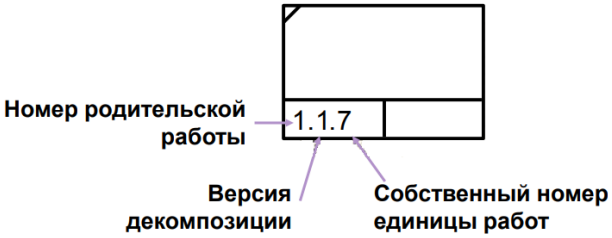
Перехрестя можуть комбінуватися для створення складних з'єднань:



Можливі множинні декомпозиції:



Номер роботи складається з номера батьківської роботи, версії декомпозиції і власного номера роботи на поточній діаграмі.



Запитання для самоперевірки

1. З яких елементів складається нотація IDEF3?
2. Які два типи діаграм використовуються в IDEF3?
3. З яких елементів складається діаграма PFDD?
4. Які типи зв’язків використовуються в діаграмі PFDD?
5. Які типи вузлів використовуються в діаграмі PFDD?

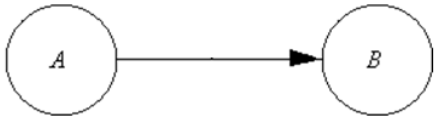
Лекція 4. Нотація IDEF3 (продовження)

1. Діаграма OSTN
2. Приклади використання нотації IDEF3

1. Діаграма OSTN

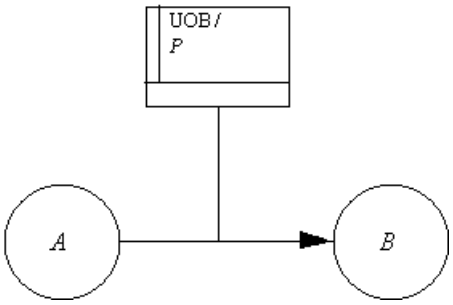
Якщо діаграми PFDD технологічний процес «З точки зору спостерігача», то інший клас діаграм IDEF3 *Діаграми переходу стану об'єкта* (OSTN) дозволяє розглядати той же самий процес «З точки зору об'єкта».

Стани об'єкту та Зміни стану є ключовими поняттями OSTN діаграми. Стани об'єкта відображаються колами, а їх зміни спрямованими лініями. Кожна лінія має посилання на відповідний функціональний блок UOB, в результаті якого відбулося відображене їй зміна стану об'єкту.



Референти (Referens) та примітки (Notes) необхідні для більш глибокого розуміння сенсу і спрощення конструкцій, тобто для полегшення їх сприйняття і усунення будь-яких варіантів невиразності або різночитань.

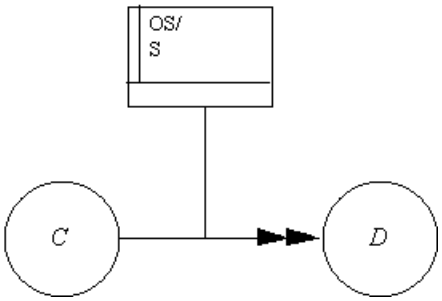
Наприклад:



Існують декілька типів таких об'єктів:

- Call and Continue Referent (Викликай та продовжуй). Використовується для виклику раніше описаного UOB без дублювання. Вказує, що в процесі виконання основного UOB необхідно буде звернутися до описаного раніше до моменту завершення поточного UOB.
- Call and Wait Referent (Викликай та чекай). Використовується для передачі управління або визначення циклу в процесі обробки. Показує, що в процесі виконання поточного UOB потрібно звернутися до описаного раніше, після чого обов'язково дочекатися його завершення, і тільки потім можна буде завершити поточний UOB.
- Note (Примітка).

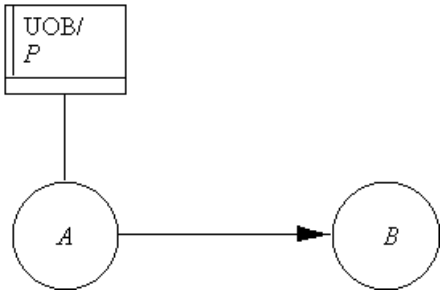
Приклад позначення «Викликай та продовжуй»



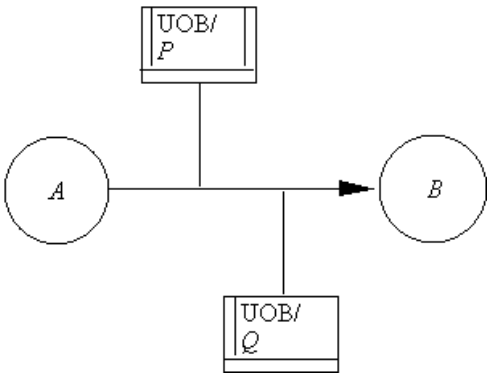
Приклад референта «Викликай та чекай».



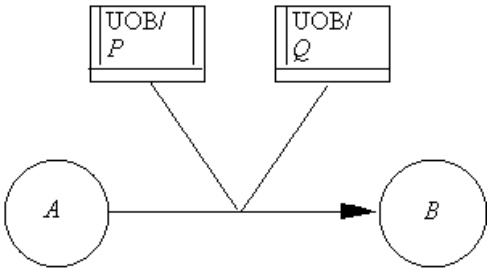
Існують референти, прикріплені до об'єктних станів. У даній ситуації досить часто використовується «утримання» об'єкта в даному стані; наприклад, в процесі заморожування дана речовина може підтримуватися в твердому стані. Ситуації цього типу можна представити конструкцією:



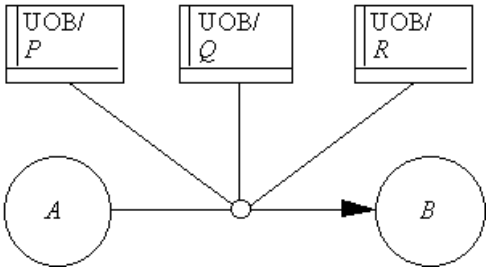
Схематики об'єктів з безліччю референтів. При переході з одного стану в інший часто може спостерігатися більш складний хід подій, в порівнянні з тим ходом подій, який може бути визначений одним референтом. Можна припускати, що деталі такого перебігу подій можуть бути забезпечені окремою схематики процесів. Для цієї мети до однієї дузі прикріплюється безліч референтів.



Схематика об'єктів з великою кількістю одночасних за часом референтів зображуються наступним чином:



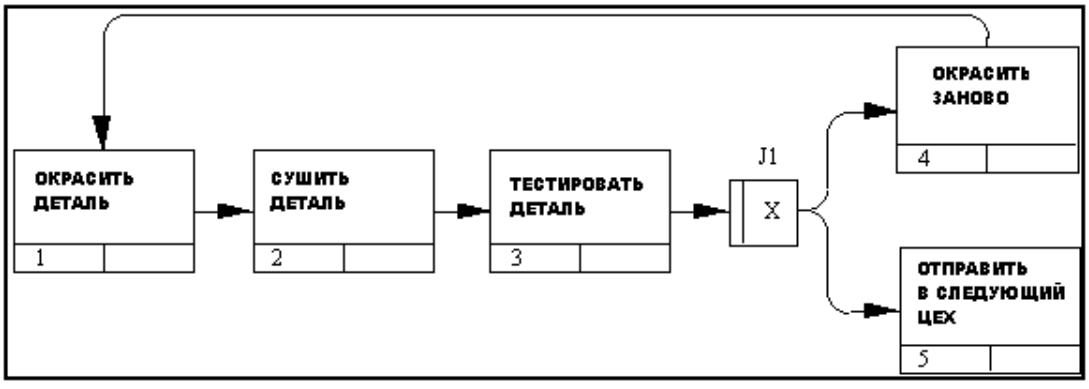
Для відображення об'єктів з невизначеними за часом референтами використання додаткового символу (маркера тимчасової невизначеності, який позначається невеликим кружком на зв'язку переходів станів) для подання переходу станів, в якому відсутня (наскільки відомо) певне упорядкування за часом UOB, включених в перехід станів. Приклад схематики об'єктів з невизначеними за часом референтами:

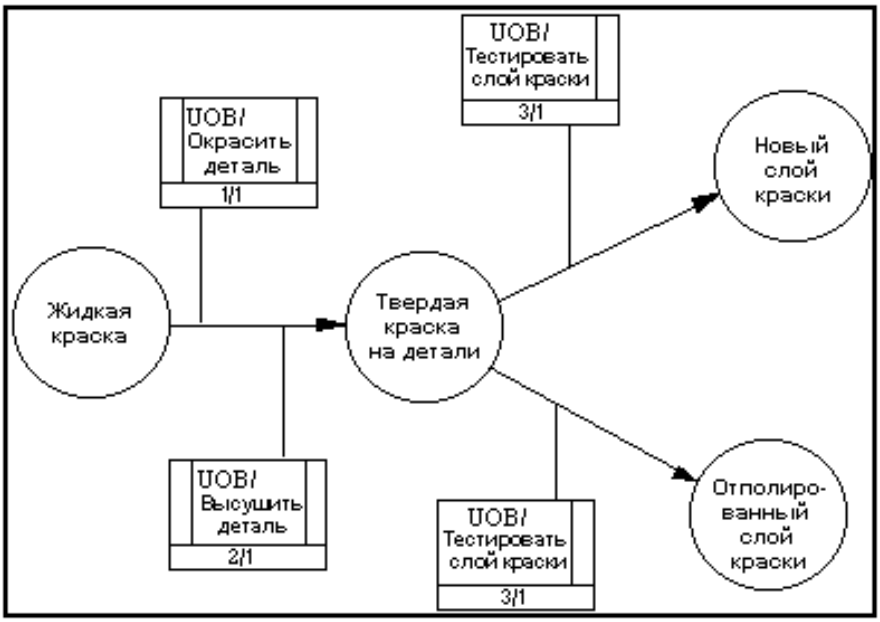


В описі референта обов'язково вказують його тип, а також мітку UOB або іншого об'єкта, з яким буде проводитися робота. Для визначення типу переходу використовується локатор.

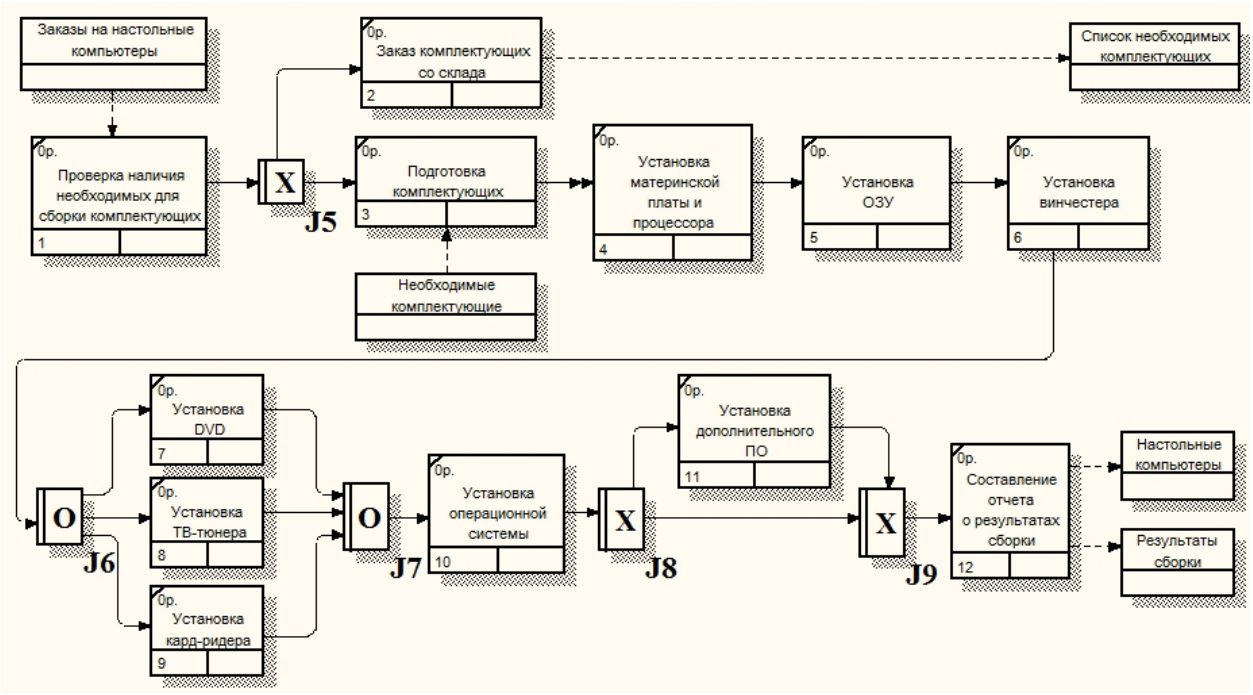
2. Приклади використання нотації IDEF3

Розглянемо приклад діаграми PFDD для завдання «Забарвлення деталі» та діаграму OSTN для цього ж завдання.





Розглянемо задачу «Збірка настільних комп'ютерів».



Запитання для самоперевірки

1. З яких елементів складається діаграма OSTN?
2. Що таке референти та примітки?

Лекція 5. Нотація IDEF1X

1. Компоненти синтаксису IDEF1X
2. Діаграма IDEF1X

1. Компоненти синтаксису IDEF1X

IDEF1X є методом для розробки реляційних баз даних і використовує умовний синтаксис, спеціально розроблений для зручного побудови концептуальної схеми.

Сутність представляє безліч реальних або абстрактних предметів (людей, об'єктів, місць, подій, станів, ідей, пар предметів тощо), що володіють загальними атрибутами або характеристиками.

Окремий елемент цієї множини називається *екземпляром сутності*. Реально існуючий об'єкт або предмет може бути представлений в декількох сутностях моделі даних.

Сутність є *незалежною від ідентифікаторів* або просто незалежною, якщо кожен екземпляр сутності може бути однозначно ідентифікований без визначення його відносин з іншими сутностями.

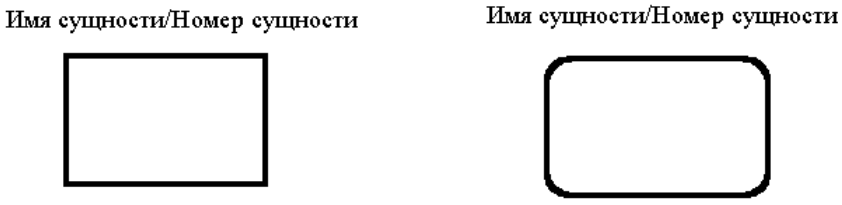
Сутність називається *залежною від ідентифікаторів* або просто залежною, якщо однозначна ідентифікація екземпляра сутності залежить від його ставлення до іншої сутності.

2. Діаграма IDEF1X

Сутність зображується прямокутником. Якщо сутність залежна від ідентифікаторів, то кути блоку закругляються. Кожній сутності присвоюється унікальне ім'я і номер, розділяються косою рисою і поміщаються над блоком. Номер суті – позитивне ціле число.

Іменем суті є граматичний оборот іменника (іменник, у якого можуть бути прикметники і прийменники), що описує сутністю. Іменник має вживатися в єдиному, а не в множині. На діаграмі сутність повинна бути представлена тільки один раз.

Наприклад:



Правила, пов'язані з сутностями :

- Кожна сутність повинна мати унікальне ім'я, і до одного і того ж імені повинна завжди застосовуватися одна й та ж інтерпретація. Одна і та ж інтерпретація не може застосовуватися до різних іменах, якщо тільки вони не є псевдонімами.
- Сутність володіє одним або декількома атрибутами, які або належать сутності, або успадковуються через ставлення.
- Сутність володіє одним або декількома атрибутами, які однозначно ідентифікують кожен екземпляр сутності.
- Кожна сутність може мати будь-яку кількість відносин з іншими сутностями моделі.
- Якщо зовнішній ключ цілком використовується в якості первинного ключа сутності або його частини, то сутність є залежною від ідентифікатора.

– Якщо використовується тільки частина зовнішнього ключа або взагалі не використовуються зовнішні ключі, то сутність є незалежною від ідентифікатора.

Атрибут представляє тип характеристик або властивостей, асоційованих з безліччю реальних або абстрактних об'єктів (людей, об'єктів, місць, подій, станів, ідей, пар предметів тощо).

Екземпляр атрибута – це певна характеристика окремого елемента множини. Екземпляр атрибута визначається типом характеристики і її значенням, званим значенням атрибута.

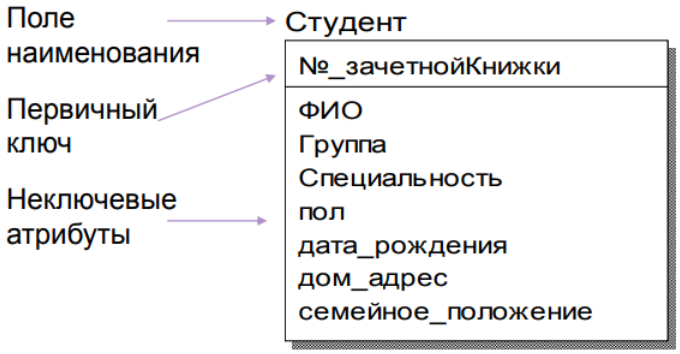
У IDEF1X – моделі атрибути асоціюються зі специфічними сутностями. Таким чином, екземпляр сутності повинен володіти єдиним певним значенням для асоційованого атрибута.

Наприклад, асоційованими з сутністю СЛУЖБОВЕЦЬ можуть бути атрибути ФАМІЛІЯ_СЛУЖБОВЦЯ та ДАТА_НАРОДЖЕННЯ. Екземпляр сутності СЛУЖБОВЕЦЬ може мати як значення атрибутів "Дженні Лінн" і "27 лютого, 1973".

Сутність повинна мати, атрибутом або комбінацією атрибутів, чиї значення однозначно визначають кожен екземпляр сутності. Ці атрибути утворюють первинний *ключ сутності*.

Кожен атрибут ідентифікується унікальним ім'ям, що виражається граматичним оборотом іменника. Іменник має бути в однині. Кожен атрибут всередині блоку сутності займає один рядок. Атрибути, що визначають первинний ключ, розміщуються нагорі списку і відділяються горизонтальною лінією.

Наприклад:



Розрізняють :

– власні,

– успадковані атрибути.

Власні атрибути є унікальними в рамках моделі.

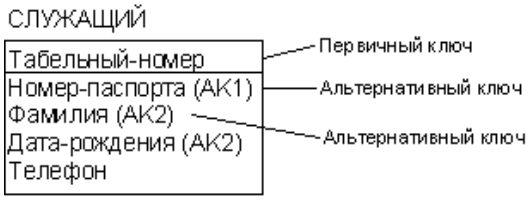
Успадковані передаються від сутності – батька при визначенні зв'язку.

Можливий ключ сутності – це один або декілька атрибутів, чиї значення однозначно визначають кожен екземпляр сутності.

Кожна сутність повинна мати хоча б один можливий ключ. У деяких випадках сутність може мати більше одного атрибута, що однозначно ідентифікує екземпляри сутності. При існуванні декількох можливих ключів один з них вибирається в якості первинного ключа, а решта – як альтернативні ключі.

Кожному альтернативному ключу привласнюється унікальний цілий номер. Цей ключ вказується за допомогою розміщення праворуч від кожного атрибута ключа укладених в дужки букв АК з номером альтернативного ключа.

Наприклад:



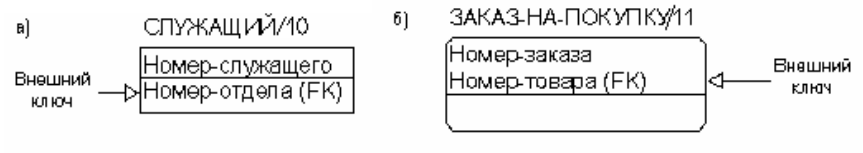
Якщо між двома сутностями є специфічне ставлення зв'язку або категоризації, то атрибути, що входять в первинний ключ батьківської або загальної суті, успадковуються як атрибути сутністю-нащадком. Ці успадковані атрибути називаються *зовнішніми ключами*.

Наслідуваний атрибут може використовуватися по суті в якості частини або цілого первинного ключа, альтернативного ключа або неключових атрибута.

Якщо всі атрибути первинного ключа сутності-батька успадковуються в якості частини первинного ключа сутності-нащадка, то відношення називається що ідентифікує. Якщо який-небудь з успадкованих атрибутів не є частиною первинного ключа, то відношення називається що не ідентифікує.

Зовнішній ключ зображується за допомогою приміщення всередину блоку сутності-нащадка імен успадкованих атрибутів, після яких слідують букви FK в дужках.

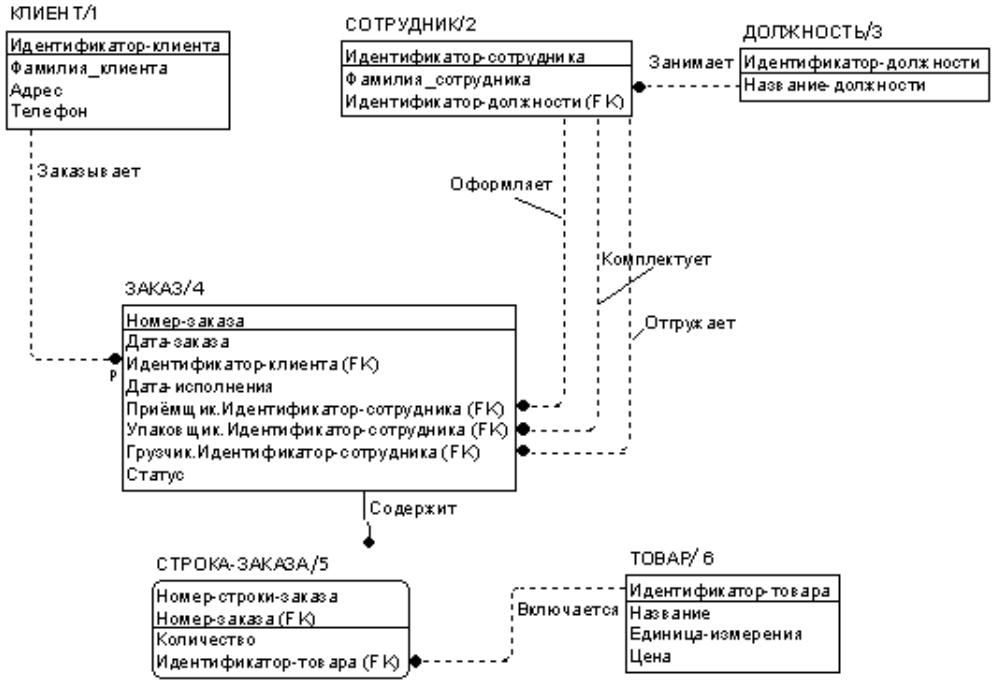
Наприклад:



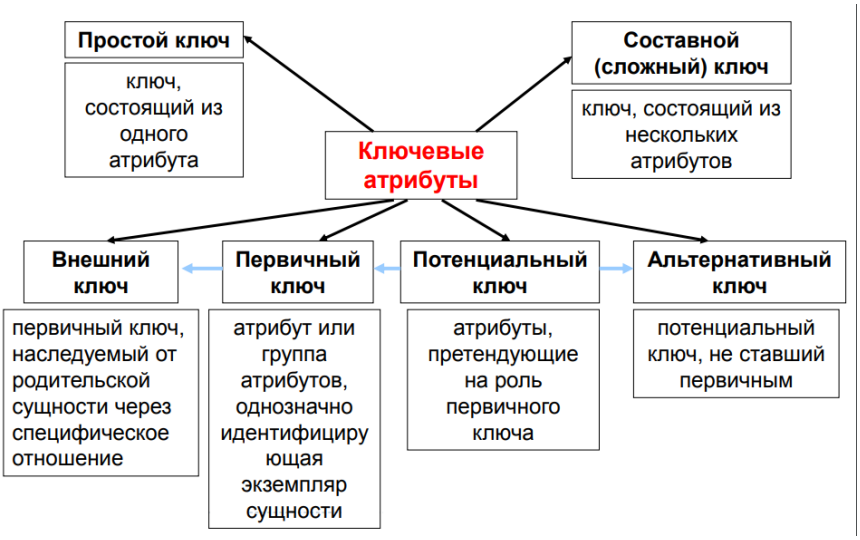
Якщо атрибут, що успадковується, не належить первинному ключу сутності-нащадка, то він зображується нижче горизонтальної лінії.

Якщо атрибут, що успадковується, належить первинному ключу сутності-нащадка, то він поміщається вище лінії, а сутність зображується блоком із закругленими кутами.

Розглянемо приклад:



Розглянуті поняття ключів можна звести у наступну схему:



Типи залежних сутностей :

– Характеристична – це залежна дочірня сутність, яка пов'язана тільки з однієї батьківської сутністю і за змістом зберігає інформацію про характеристики батьківської сутності.
– Категоріальна – це дочірня сутність в ієрархії успадкування
– Асоціативна – сутність, пов'язана з декількома батьківськими сутностями. Така сутність містить інформацію про зв'язки сутності.
– Що іменує – окремий випадок асоціативної суті, не має власних атрибутів, тільки атрибути батьківського сутності.

Специфічне ставлення зв'язку або просто відношення зв'язку, зване також ставлення батько-нащадок, це асоціація або зв'язок між сутностями, при якій кожен екземпляр однієї сутності, званої батьківської сутністю, асоційований з довільною (в тому числі нульовим) кількістю примірників другої сутності, званої сутністю – нащадком.

Кожен екземпляр сутності-нащадка асоційований в точності з одним екземпляром сутності-батька. Таким чином, примірник сутності-нащадка може існувати тільки при існуванні сутності-предка.

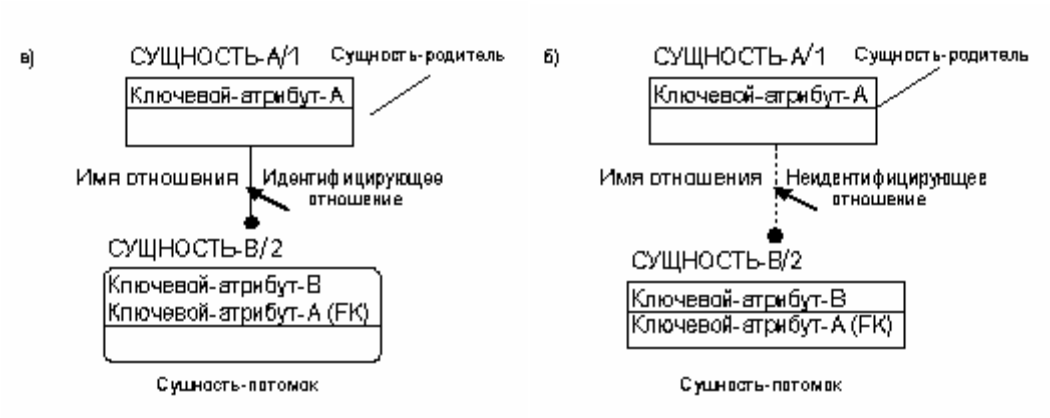
Якщо екземпляр сутності – нащадка однозначно визначається своїм зв'язком з сутністю-батьком, то відношення називається *відношенням, що ідентифікує*.

Наприклад, якщо з кожним проектом пов'язано одне або більше завдань і завдання однозначно ідентифікуються тільки в межах свого проекту, то між сутностями ПРОЕКТ і ЗАВДАННЯ буде існувати відношення що ідентифікує.

Якщо кожен екземпляр сутності – нащадка може бути однозначно ідентифікований без знання пов'язаного з ним екземпляра сутності-батька, то відношення називається *відношенням, що не ідентифікує*.

Наприклад, хоча між сутностями ПОКУПЕЦЬ і ЗАМОВЛЕННЯ_НА_ ПОКУПКУ може існувати відношення залежного існування, замовлення на покупку можуть однозначно ідентифікуватися номером замовлення на покупку без ідентифікації асоційованого покупця.

Приклад позначення:



Специфическое відношення зв'язку зображується лінією, що проводиться між сутністю-батьком і сутністю-нащадком з точкою на кінці лінії у сутності-нащадка. Потужність за замовчуванням – «нуль, один або багато».

Буква P (positive) означає потужність «один або багато» і поміщається близько точки.

Буква Z (zero), вміщена близько точки, означає потужність «нуль або один».

Якщо потужність в точності дорівнює деякому числу N, це число (ціле, позитивне) поміщається близько точки.

Приклад позначення:



Відношенню дається ім'я, виражене граматичним оборотом дієслова і поміщається біля лінії відносини. Ім'я кожної відносини між двома сутностями повинно бути унікальним, але імена відносин в моделі не обов'язково повинні бути унікальними. Ім'я відносини в більшості випадків формується з точки зору батьків.

Наприклад, твердження «Проект складається з одного або більше завдань» може бути виведено з відносини, що зображує ПРОЕКТ, як сутності – батька, ЗАВДАННЯ – в якості сутності – нащадка з символом потужності P, СКЛАДАЄТЬСЯ_З – як ім'я відносини.

Приклад асоціативного відношення:



Деякі існуючі об'єкти є категоріями інших реально існуючих об'єктів, тому сутності можуть бути категоріями інших сутностей. Наприклад, для загальної суті службовець можна вказати дві сутності-категорії: ШТАТНИЙ_СЛУЖБОВЕЦЬ і СЛУЖБОВЕЦЬ_ПОГОДИННИК

Розрізняють повну і неповну категоризацію.

Відношення повної категоризації – відношення між двома або більше сутностями, в якому кожен екземпляр однієї сутності, званої загальної сутністю, пов'язаний в точності з одним екземпляром однієї і тільки однієї з інших сутностей, званих сутностями-категоріями.

Якщо існує екземпляр загальної суті, не пов'язаний ні з яким примірником з сутностей-категорій то таке ставлення називається відношенням *неповної категоризації*.

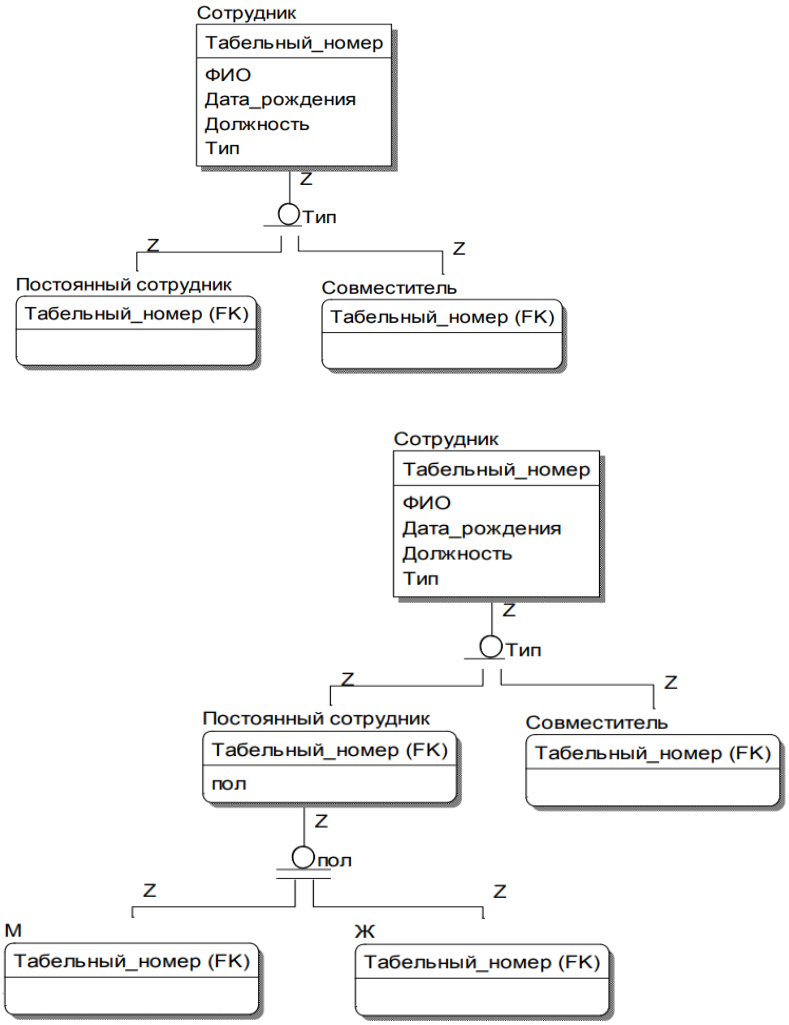
Відношення категоризації зображується лінією, що веде з загальної суті до підкресленому колу. Окремі лінії ведуть від підкресленого кола до кожної з сутностей-категорій.

Якщо коло підкреслено двічі  , це вказує на повноту безлічі сутностей-категорій.

Якщо коло підкреслено один раз  , це вказує на неповноту безлічі категорій.

Ім'я атрибута загальної суті, використовується в якості дискриминатора, записується поряд з колом.

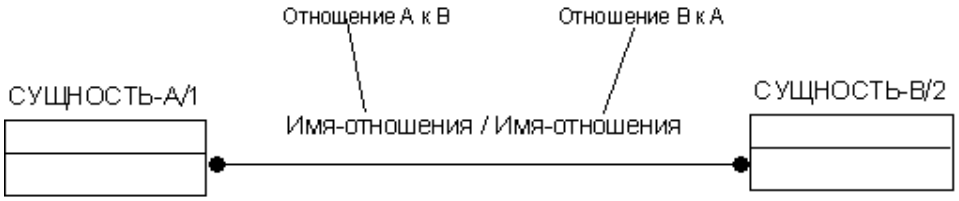
Наприклад:



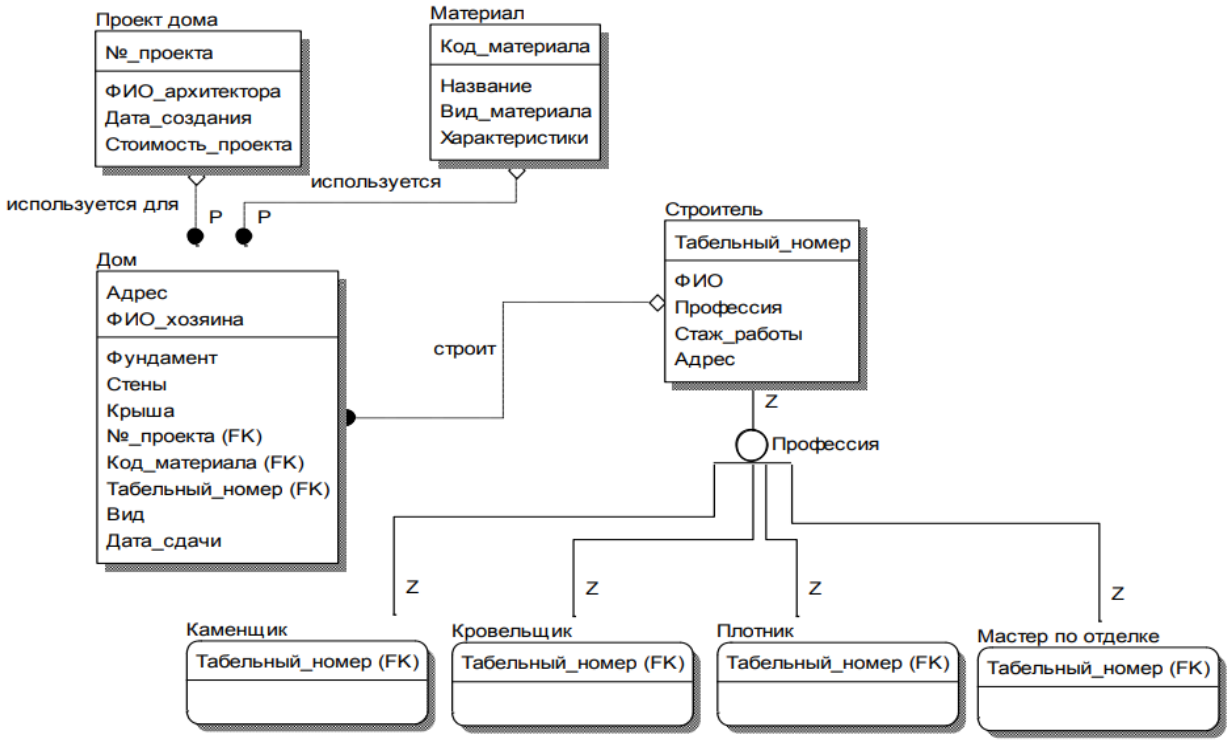
Неспецифічне відношення, зване також ставленням багато-до-багатого, це зв'язок між двома сутностями, при якій кожен екземпляр першої суті пов'язаний з довільним (в тому

числі і нульовим) кількістю екземплярів другої сутності, а кожен екземпляр другої сутності пов'язаний з довільною (в тому числі і нульовим) кількістю екземплярів першої сутності

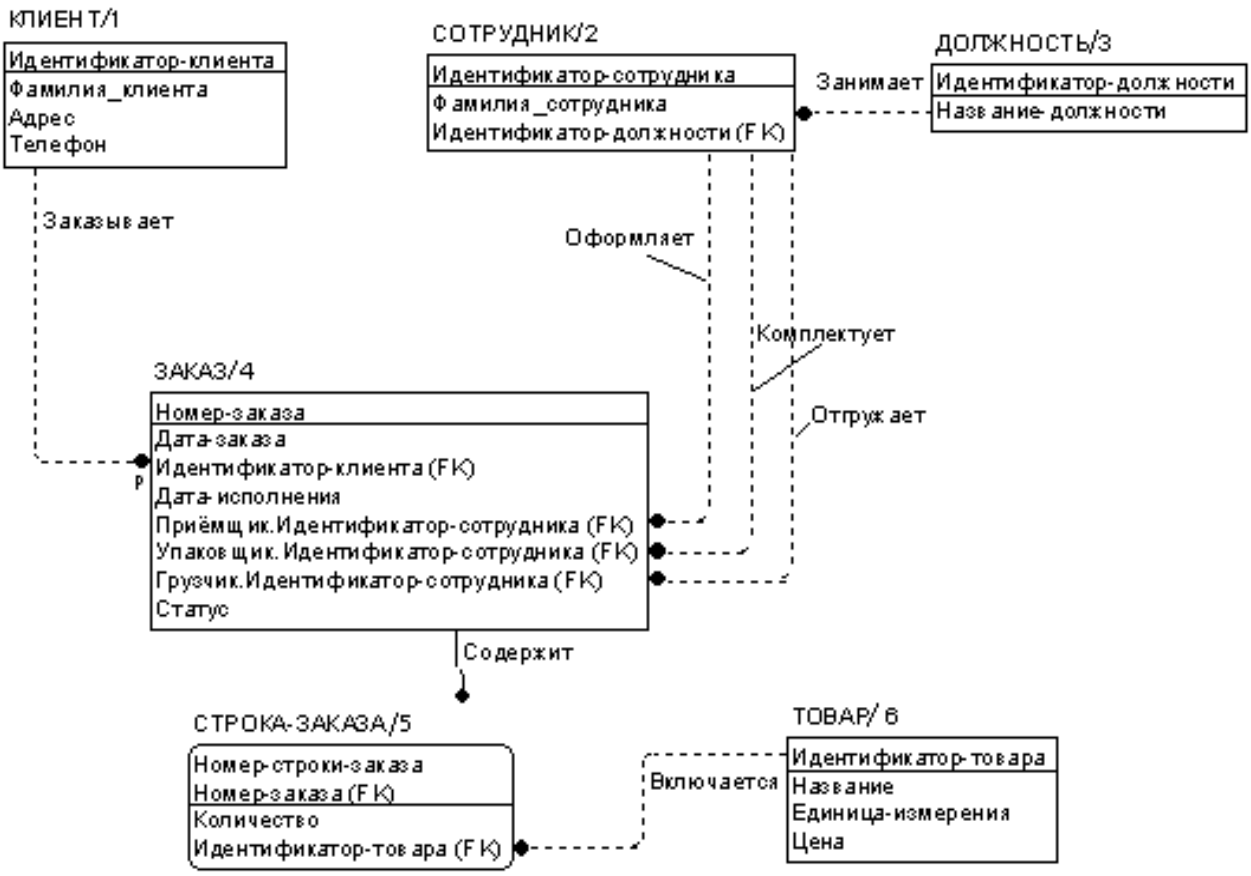
Наприклад:



Розглянемо приклад. Побудова інформаційної моделі процесу спорудження будиночка. Нехай маємо список потенційних сутностей: Будинок, Дах, Матеріали, Проект будинку, Стіни, Фундамент, Будівельники, Каменярі, Теслі, Покрівельники, Майстри по обробці. Тоді маємо наступний опис цього прикладу:



Розглянемо приклад. Клієнти деякої фірми замовляють товари. На товари, що замовляються співробітник фірми оформляє замовлення. У замовленні в загальному випадку міститься кілька рядків (позицій), в яких зазначаються відомості про товари, що замовляються та їх кількості. Інший співробітник фірми комплектує замовлення і виконує операцію упаковки. Ще один співробітник фірми виконує операцію навантаження і доставки товарів клієнту. Кожен із співробітників фірми займає певну посаду. Тоді маємо наступний опис цього прикладу:



Запитання для самоперевірки

1. З яких елементів складаються діаграми IDEF1X?
2. Що таке ключові атрибути?
3. Які види ключових атрибутів ви знаєте?
4. Які типи залежних сутностей ви знаєте?

Лекція 6. Нотація UML. Структурне моделювання

1. Модель класів
2. Компоненти синтаксису моделі класів

1. Модель класів

Моделювати систему краще з трьох різних точок зору, пов'язаних між собою. Кожна модель описує важливі аспекти системи, але для більш повного опису потрібно все три моделі:

– модель класів
– модель стану
– модель взаємодії

Модель класів становить статичні, структурні аспекти системи, пов'язані з даними.

Модель стану представляє тимчасові, поведінкові, управлінські аспекти системи.

Модель взаємодії представляє кооперацію окремих об'єктів, тобто всі аспекти системи, пов'язані з взаємодіями.

Розглянемо детальніше модель класів.

Класи – найбільш важливі будівельні блоки будь-якої об'єктно-орієнтованої системи.

Клас – це опис безлічі об'єктів з однаковими атрибутами, операціями, зв'язками і семантикою.

Моделювання системи включає в себе ідентифікацію сутностей, які важливі для конкретного уявлення. Ці сутності формують словник системи, яка моделюється. Кожна з цих сутностей відрізняється від іншої і має набір певних властивостей.

Наприклад, при побудові будинку вам як майбутньому власникові не байдуже, якими будуть стіни, двері, вікна, кабінети, системи освітлення та ін. Стіни мають якусь висоту, ширину і є суцільними. Двері характеризуються тими ж ознаками, але їх ще й відрізняється специфічною поведінкою: вони відкриваються в одну сторону. Вікна частково схожі з дверима, оскільки теж утворюють отвір в стіні, але їх функціональність неоднакова. Окремі стіни, двері і вікна рідко існують самі по собі, тому потрібно також розглянути, як конкретні екземпляри цієї суті поєднуються один з одним.

В UML всі ці сутності моделюються класами. *Клас* – це абстракція сутності, що є частиною словника системи. Клас – не індивідуальний об'єкт, а уявлення чималої кількості об'єктів: скажімо, можна концептуально представляти «стіну» як клас об'єктів з низкою загальних властивостей (висота, довжина, товщина, чи є стіна несучої тощо).

Наприклад:

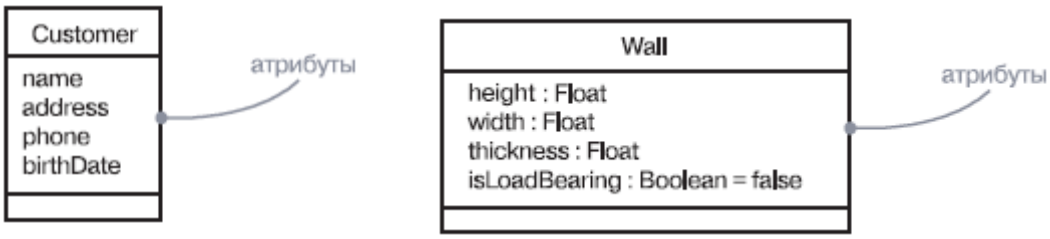


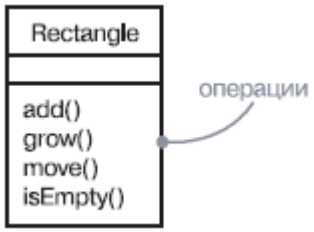
Рисунок 11.1 – Приклади класів

2. Компоненти синтаксису моделі класів

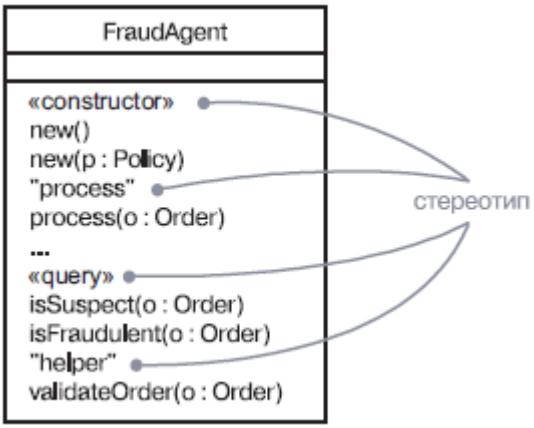
Операція (operation) – це реалізація послуги, яка може бути запрошена у будь-якого об'єкта даного класу, щоб викликати певну його поведінку. Іншими словами, операція – це

абстракція чогось, що ви можете зробити з конкретним об'єктом і взагалі з усіма об'єктами даного класу.

Клас може мати будь-яке число операцій або не мати жодної. Графічно операції представлені в розділі списку, наведеного під атрибутами класу. Допускається зазначення тільки імен операцій. Наприклад:



Щоб краще організувати довгі списки атрибутів і операцій, бажано забезпечити префіксом (ім'ям стереотипу) кожну категорію в них.

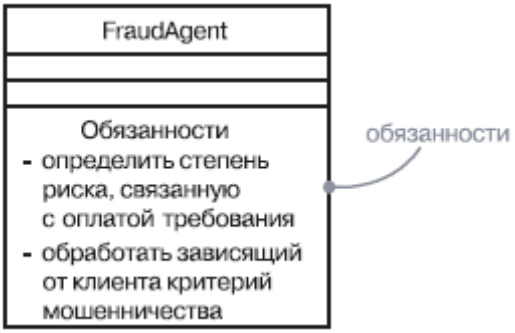


Обов'язок (responsibility) – це угода або зобов'язання класу. Коли ви створюєте клас, висувається припущення, що всі його об'єкти характеризуються однаковим станом і однаковою поведінкою.

На більш абстрактному рівні відповідні атрибути і операції є просто засобами, завдяки яким клас виконує свої обов'язки.

Наприклад, клас Wall (Стіна) відповідає за інформацію щодо висоти, ширини, товщини; клас FraudAgent (Агент За запобігання шахрайству), який можна зустріти в додатку, що обробляє кредитні карти, - за обробку запитів і визначення їх обґрунтованості, підозрілості або незаконність.

У класу може бути скільки завгодно обов'язків, хоча на практиці кожен добре структурований клас має як мінімум один обов'язок і як максимум – невеликий їх набір. Графічно обов'язки можуть бути представлені в спеціально відведеному для них розділі, в нижній частині піктограми класу.



Класи рідко існують самі по собі. Коли будуються моделі, то, як правило, фокусується увагу на групах класів, що взаємодіють один з одним. В UML такі спільноти класів формують кооперації і зазвичай візуалізують в *діаграмах класів*.

При моделюванні системи необхідно не тільки ідентифікувати сутності, що формують її словник, а й змоделювати відносини один до одного. В об'єктно-орієнтованому моделюванні існують три види зв'язків між класами, які найбільш важливі: залежність, узагальнення, асоціації.

Залежність представляє зв'язки використання між класами (включаючи уточнення, трасування і зв'язування).

Узагальнення, яке пов'язує узагальнені класи з їх спеціалізаціями.

Асоціації, що описують структурні зв'язки об'єктів.

Кожна з цих різновидів представляє окремий спосіб комбінування абстракцій.

Розглянемо зв'язки щодо метафори будівництва будинку. Стіни, двері, вікна, вбудовані шафи і системи освітлення формують частину словника системи. Всі ці об'єкти (стіни, двері, вікна, шафи та освітлювальні прилади) в сукупності формують більш високорівневі сутності – кімнати.

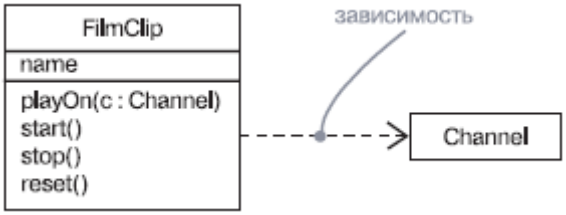
Залежність являє собою зв'язок використання. Наприклад, труби залежать від водонагрівача для підігріву води, яка по ним передається.

Асоціація – це структурний зв'язок між екземплярами. Наприклад, кімнати складаються зі стін та інших об'єктів; в стіни вмонтовані двері і, можливо, вікна; через стіни можуть тягнутися труби.

Узагальнення пов'язує узагальнені класи з більш спеціалізованими і тому відомі як зв'язки успадкування («клас-підклас», або «батько-нащадок»). Наприклад, вітраж – це вікно з дуже великими, жорстко фіксованими панелями; патіо – різновид вікна, що відкривається убік.

Залежність (dependency) – це зв'язок, який встановлює, що одна сутність, наприклад клас Window (Вікно), використовує інформацію і сервіс (операцію або послугу), що подаються іншою сутністю, наприклад класом Event (Подія), але не обов'язково – навпаки.

Залежність зображується у вигляді пунктирної лінії зі стрілкою, спрямованої на залежну сутність.



Вибирається залежність, коли потрібно показати, що одна сутність використовує іншу. Найчастіше цей тип зв'язку застосовується для того, щоб показати, що один клас

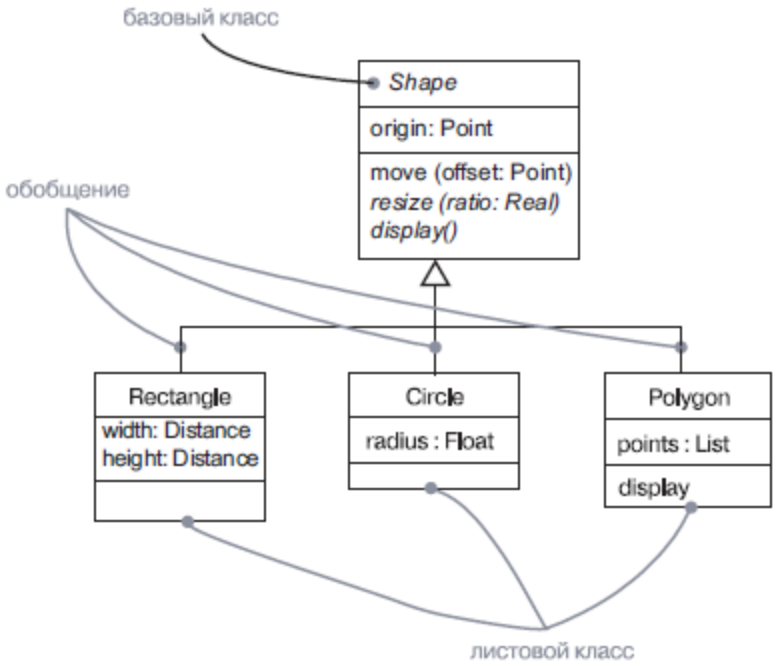
використовує операції іншого класу (або використовує змінні чи аргументи типу іншого класу).

Узагальнення (generalization) – це зв'язок між сутністю загального характеру (званої суперкласом, або батьком) і більш специфічною сутністю (званої подклассом, дочірнім класом або нащадком). Іноді узагальнення називають зв'язком типу «є». Графічно узагальнення представлено суцільною лінією зі стрілкою в формі великого порожнього трикутника, що вказує на батька. Використовуйте узагальнення, коли необхідно зобразити зв'язок «батько-нащадок».

Клас, що не має батьків, але має одну або кількох нащадків, називається *кореневим* (root) або *базовим*. Клас, що не має нащадків, називається *листовим* (leaf).

Про клас, у якого є тільки один батько, кажуть, що він використовує *одиночне спадкоємство*, на відміну від класу, у якого більш ніж один батько – *множинне спадкування*.

Наприклад:



Асоціація – це структурний зв'язок, який вказує, що об'єкти однієї сутності з'єднуються з об'єктами іншої. Так, маючи асоціацію між двома класами, можна з'єднати об'єкти одного класу з об'єктами іншого.

Цілком припустимо, щоб обидва кінці асоціації з'єднували один і той же клас – іншими словами, один об'єкт класу може зв'язуватися з іншим об'єктом того ж класу. Асоціація, що зв'язує два класи, називається *бінарною*. Зустрічаються так звані *n-арні асоціації*, які з'єднують більше двох класів. Графічно асоціація представлена суцільною лінією, два кінці якої з'єднують один або різні класи.



Асоціація може мати ім'я, яке використовується для опису природи зв'язку. Тому значення імені не повинно бути двозначним. Використовуючи стрілочку в формі трикутника, ви можете вказати напрямок, в якому слід читати це ім'я.

Коли клас бере участь в асоціації, він виконує в зв'язку з цим конкретну роль. *Роль* – це «обличчя» класу, яка знаходиться на дальньому кінці асоціації, представлене класу, що знаходиться на її ближньому кінці. Можна явно іменувати роль, яку виконує клас в асоціації. Роль, яку відіграє клас, що знаходиться на кінці асоціації, називається *кінцевим ім'ям*.

Наприклад, клас Person (Людина), який грає роль employee (працівник), асоційований з класом Company (Компанія), що грає роль employer (роботодавець).



У багатьох ситуаціях моделювання важливо знати, скільки об'єктів може бути з'єднане одним екземпляром асоціації. Цей параметр називається множинністю ролі асоціації. *Множинність* (multiplicity) представляє діапазон цілих чисел, який вказує можливу кількість пов'язаних об'єктів.

Множинність може бути визначена як одиниця (1), нуль або один (0..1), багато (0 .. *), один або кілька (1 .. *). Можна задавати діапазон цілих значень, наприклад 2..5, або встановлювати точне число, наприклад 3 (еквівалент записи 3..3).

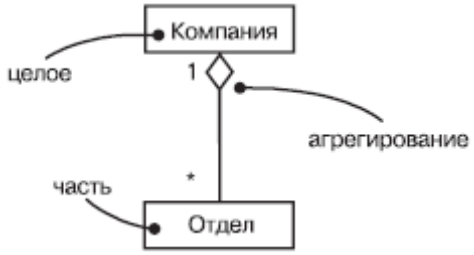
Наприклад, кожна компанія може наймати одного або декількох осіб (множинність 1 .. *); кожній людині зіставлено 0 або більше компаній-роботодавців (множинність * – еквівалент записи 0 .. *).



Проста асоціація між двома класами представляє структурну зв'язок між рівноправними елементами: обидва класу концептуально знаходяться на одному рівні – жоден з них не може вважатися важливіше іншого.

Можна також змоделювати зв'язок «ціле-частина», в якій один клас представляє велику сутність (ціле), що містить в собі більш дрібні (частини). Цей тип зв'язків, заснованих на відносинах володіння, називається *агрегацією* і має на увазі, що об'єкт-ціле володіє об'єктами-частинами.

По суті, *агрегація* – це особливий вид асоціації, тому зображується вона лінією простої асоціації, до якої доданий порожній ромб з боку об'єкта-цілого.



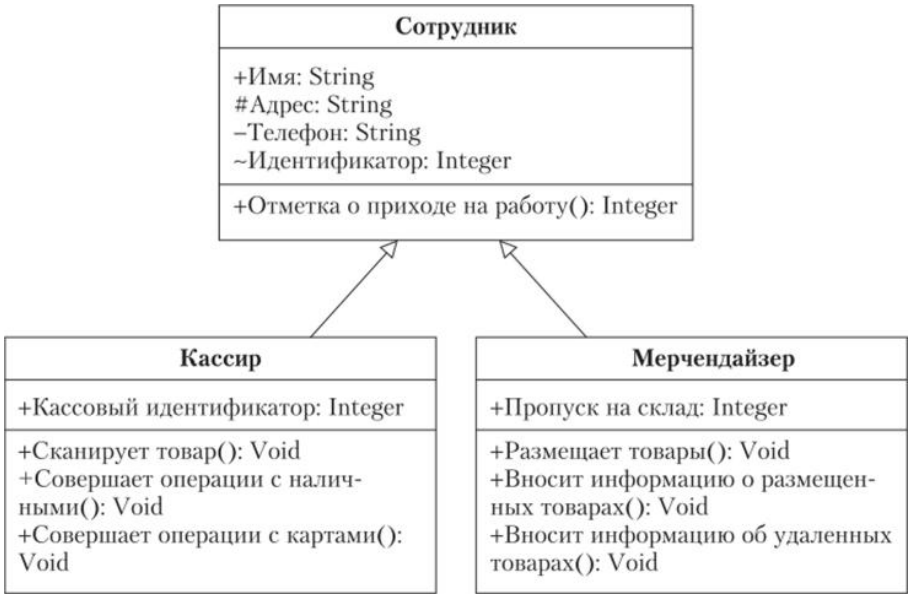
Найпоширеніший вид залежності – це зв'язок класу, який використовує інший клас в якості параметра своєї операції. Щоб змоделювати цей зв'язок використання, необхідно створити залежність, спрямовану від класу з операцією до класу, що використовується в якості її параметра.

Наприклад:

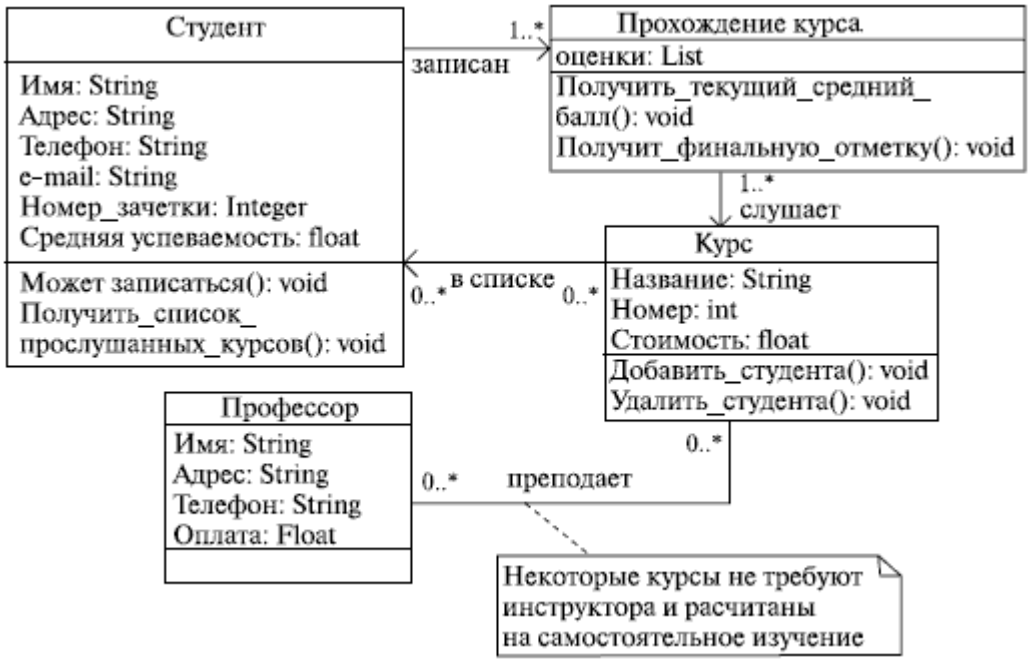


Розглянемо приклади діаграм класів.

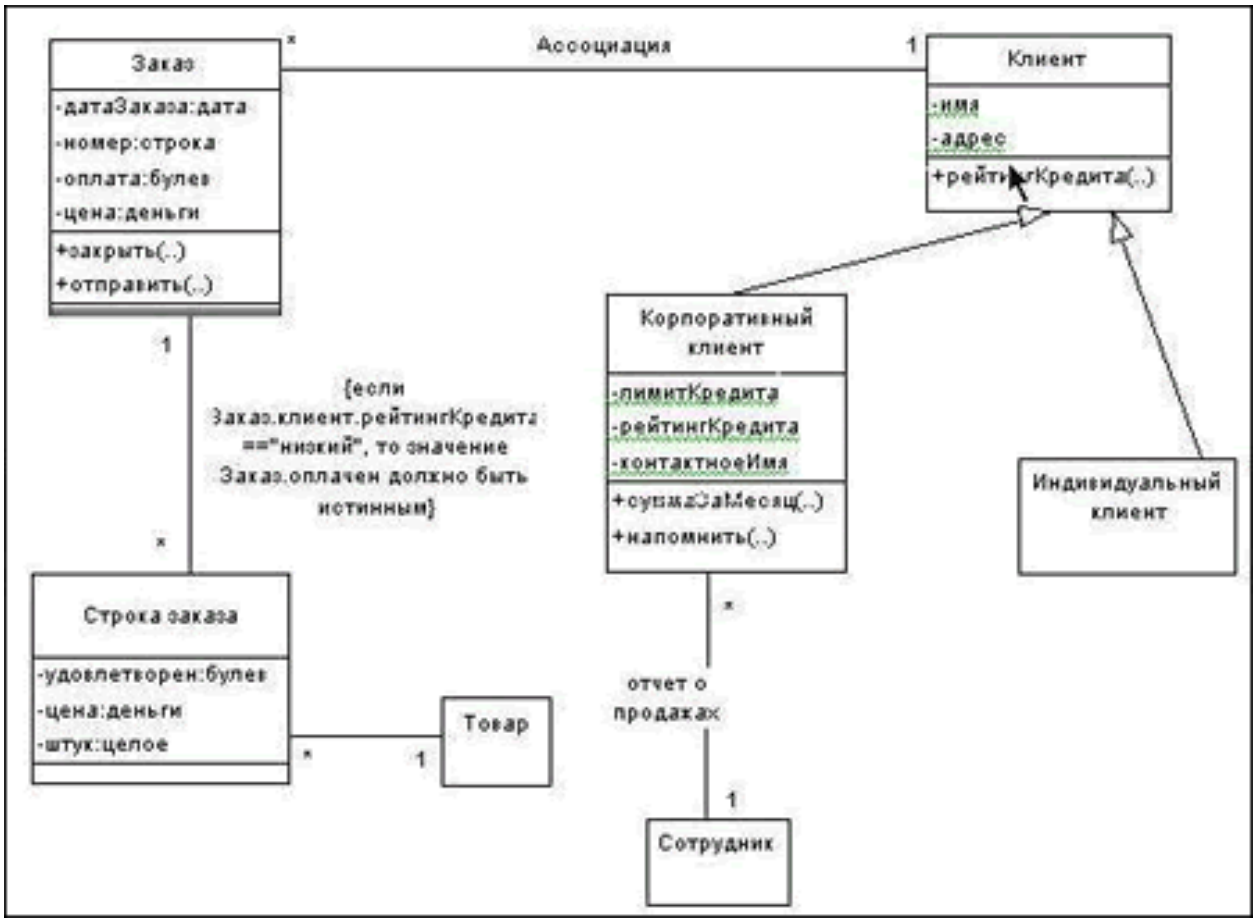
Приклад 1.



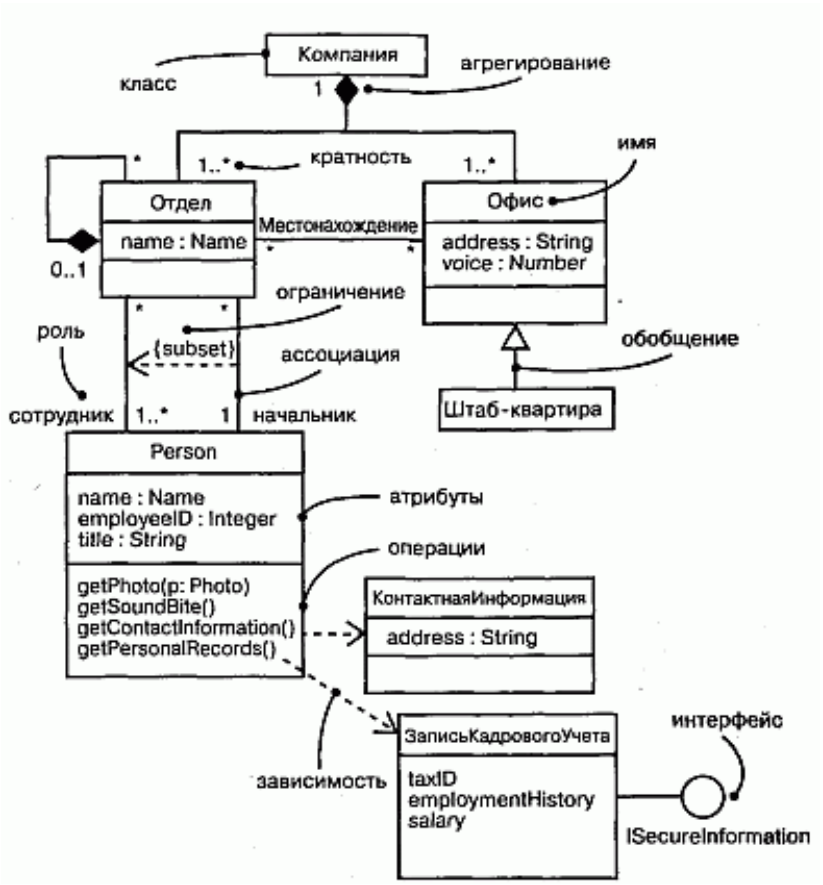
Приклад 2.



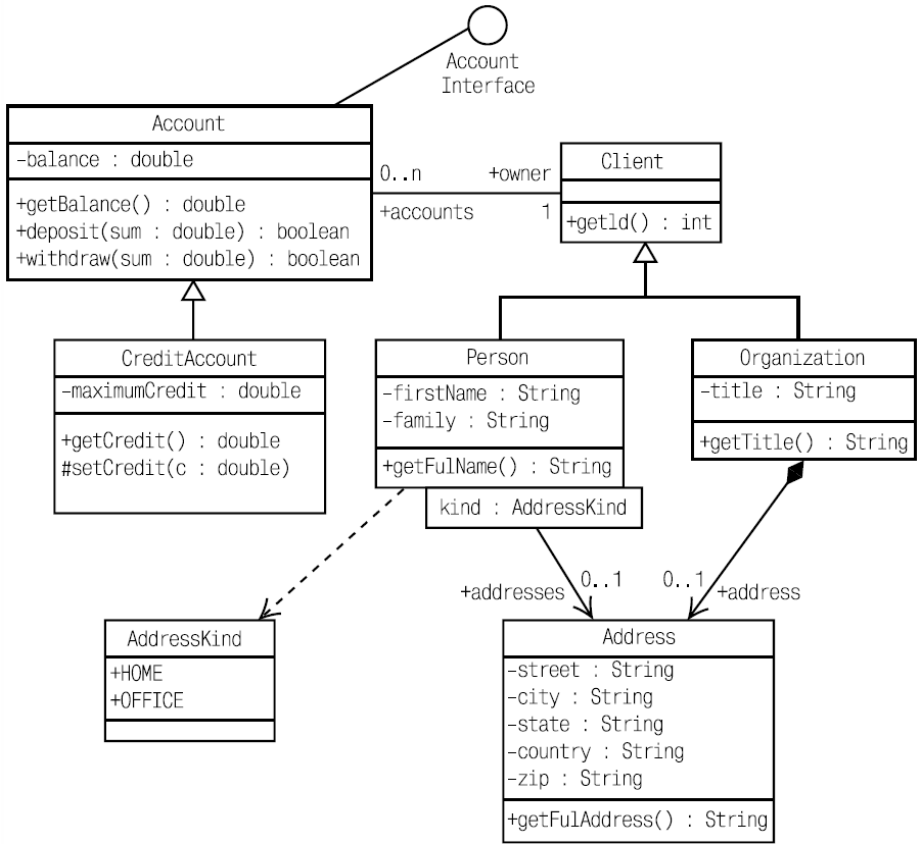
Приклад 3.



Приклад 4.


Компания
класс
агрегирование
1
1..*
кратность
1..*
имя
Отдел
name : Name
Местонахождение
Офис
address : String
voice : Number
0..1
ограничение
роль
{subset}
ассоциация
обобщение
сотрудник
1..*
1
начальник
Штаб-квартира
Person
name : Name
employeeID : Integer
title : String
атрибуты
операции
getPhoto(p: Photo)
getSoundBite()
getContactInformation()
getPersonalRecords()
КонтактнаяИнформация
address : String
ЗаписьКадровогоУчета
taxID
employmentHistory
salary
интерфейс
зависимость
ISecureInformation

Приклад 5.


Account
Interface
Account
-balance : double
+getBalance() : double
+deposit(sum : double) : boolean
+withdraw(sum : double) : boolean
0..n
+owner
+accounts
1
Client
+getId() : int
CreditAccount
-maximumCredit : double
+getCredit() : double
#setCredit(c : double)
Person
-firstName : String
-family : String
+getFulName() : String
kind : AddressKind
Organization
-title : String
+getTitle() : String
+addresses
0..1
0..1
+address
AddressKind
+HOME
+OFFICE
Address
-street : String
-city : String
-state : String
-country : String
-zip : String
+getFulAddress() : String

Запитання для самоперевірки

1. Для чого використовується нотація UML?
2. Що таке клас?
3. Що таке атрибути? Для чого вони використовуються?
4. Що таке операції? Для чого вони використовуються?
5. Що таке обов'язок? Для чого він використовуються?
6. Які три види зв'язків між класами ви знаєте?
7. Коли використовується кожен вид зв’яку?

Лекція 7. Нотація UML. Моделювання поведінки

1. Моделі стану та взаємодії
2. Діаграми станів

1. Моделі стану та взаємодії

Варіанти використання застосовуються для вираження необхідної поведінки системи, що розробляється, без опису реалізації цієї поведінки. Вони дозволяють розробникам, кінцевим користувачам і експертам в предметної області досягти взаєморозуміння, а крім того, допомагають упевнитися в правильності архітектурних рішень і перевіряти систему по ходу її розробки.

В UML поведінка моделюється за допомогою варіантів використання, що специфікуються незалежно від реалізації.

Варіант використання – це опис безлічі послідовних дій (включаючи варіації), які виконуються деяким суб'єктом з метою отримання результату, значимого для деякої діючої особи.

На системному рівні варіант використання описує набір послідовностей, кожна з яких представляє взаємодію сутностей, що знаходяться поза системою (діючих осіб), з самою системою і її ключовими абстракціями.

Діюча особа представляє собою логічно пов'язану множину ролей, які грають користувачі системи під час взаємодії з нею. Діючими особами можуть бути як люди, так і автоматизовані системи.

Будь-який варіант використання повинен виконувати певний обсяг роботи. З точки зору діючої особи він робить щось, що представляє певну цінність: наприклад, обчислює результат, створює новий об'єкт або змінює стан іншого об'єкту.

Можна застосовувати варіанти використання до всієї системи або до її частинам, в тому числі до підсистем і навіть до індивідуальних класів і інтерфейсів.

У кожному разі варіанти використання не тільки представляють бажану поведінку цих елементів, але також можуть служити основою сценаріїв тестування на різних етапах розробки.

Варіанти використання та діючі особи в UML зображуються, як показано на рисунку 7.1.



Рисунок 7.1 – Приклад зображення варіанта використання та діючій особи

Варіант використання (use case) – це опис безлічі послідовних дій, включаючи їх варіанти, які виконуються системою з метою отримання значущого результату для діючої особи. Зображується у вигляді еліпса.

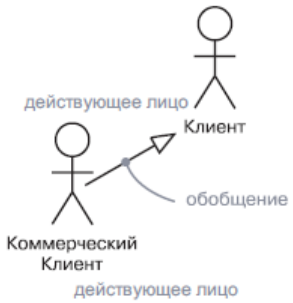
Суб'єкт – це клас, описаний набором варіантів використання. Зазвичай мова йде про систему або підсистему. Варіанти використання представляють аспекти поведінки класу. Діючі особи ж представляють аспекти інших класів, взаємодіючих з суб'єктом. Взяті разом, варіанти використання описують повну поведінку суб'єкта.

Кожен варіант використання повинен мати ім'я, що відрізняє його від інших. Ім'я варіанта використання являє собою текстовий рядок і називається *простим ім'ям*. До *кваліфікованого імені* додається префікс – ім’я пакета, в якому знаходиться варіант використання. Зазвичай при зображенні варіанту використання вказується тільки його ім’я. Наприклад:



Діюча особа представляє собою пов'язану безліч ролей, які виконують користувачі варіантів використання під час взаємодії з ними. Зазвичай діюча особа представляє ту роль, яку в даній системі грає людина, апаратний пристрій або навіть інша система.

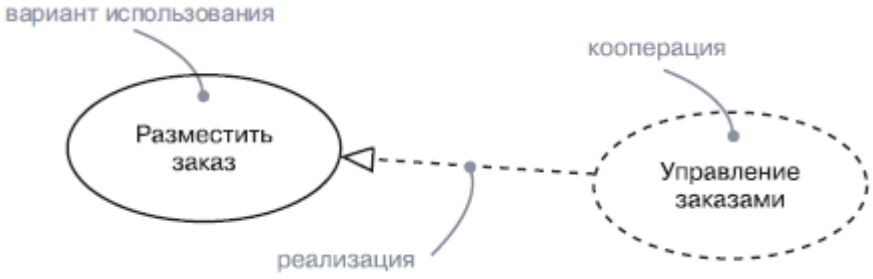
Діючі особи зображуються у вигляді людських фігурок. Можна вводити загальні типи діючих осіб, такі як Customer, і спеціалізувати їх (наприклад, створити різновид CommercialCustomer – комерційний клієнт), визначивши зв'язок узагальнення.



Діючі особи можна пов'язувати з варіантами використання тільки за допомогою асоціацій. Асоціація між дійовою особою і варіантом використання показує, що вони спілкуються один з одним, можливо, посилаючи або приймаючи повідомлення.

Необхідно реалізувати варіанти використання шляхом створення спільнот класів та інших елементів, які працюють разом для реалізації поведінки, описаного варіантом використання. Така спільнота елементів, що володіє як статичної, так і динамічної структурою, моделюється в UML як *кооперація*.

На наступному рисунку показано, що реалізацію варіанту використання можна специфікувати явно через кооперацію.



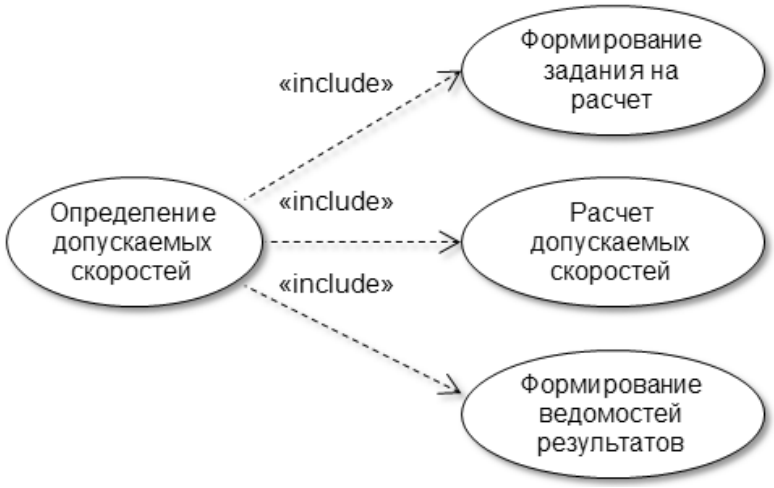
Для організації варіантів використання їх групують в пакети так само, як класи. Крім того, можна організувати варіанти використання, визначивши між ними зв'язки узагальнення, включення та розширення.

Ці зв'язки застосовуються для того, щоб виділити деяку загальну поведінку (витягуючи його з інших варіантів використання), а також різновиди (розміщуючи таку поведінку в інші варіанти використання, які розширюють даний).

Узагальнення між варіантами використання подібні узагальненням між класами. Це означає, що дочірній варіант використання успадковує поведінку і суть батьківського варіанту використання; нащадок може додати або перевизначити поведінку батьків, а крім того, бути підставленим замість нього в будь-якому місці, де той з'являється.

Зв'язок включення між варіантами використання означає, що базовий варіант використання в певному місці явно включає в себе поведінку деякого іншого. Включений варіант використання не існує окремо: він є екземпляром тільки всередині базового, який його містить. Можна вважати, що базовий варіант використання запозичує поведінку включаємого. Зв'язок включення зображується як залежність зі стереотипом include.

Наприклад:



Зв'язок розширення між варіантами використання означає, що базовий неявно включає поведінку деякого іншого в непрямо зазначеному місці. Базовий варіант використання здатний існувати окремо, але за деяких умов його поведінка може бути розширена поведінкою іншого варіанту використання.

Базовий варіант використання можна розширити лише викликом з певної точки, – так званої *точки розширення* (extension point). Щоб наочно представити ситуацію, можна уявити, що розширюючий варіант використання «заштовхує» поведінку в базовий. Зв'язок розширення зображується як залежність зі стереотипом extend. У додатковій секції можна перерахувати точки розширення базового варіанту використання. Ці точки розширення – прості мітки, які можуть з'являтися в потоці базового варіанту використання.

Наприклад:



Щоб змоделювати поведінку елемента, необхідно:

1. Ідентифікувати діючі особи, які взаємодіють з елементом. Кандидати на включення в цю групу - ті, хто потребує певної поведінки елемента для виконання своїх власних завдань, або ті, хто прямо або побічно задіяно у функціонуванні елементу.
2. Організувати діючі особи, визначивши загальні і більш спеціалізовані ролі.
3. Розглянути основні шляхи взаємодії кожної діючій особи з елементом, а також самі взаємодії, які змінюють стан елемента або його оточення або забезпечують реакцію на деяку подію.
4. Розглянути виняткові шляхи взаємодії кожної діючої особи з елементом.
5. Організувати поведінку, виявлену на етапах 3 та 4, у вигляді варіантів використання, застосовуючи зв'язки включення та розширення, щоб виділити загальну поведінку і відокремити виняткове.

Наприклад, система роздрібної торгівлі повинна взаємодіяти з замовниками, які розміщують замовлення і відстежують їх виконання.



Діаграми варіантів використання – це один з видів діаграм UML, призначених для моделювання динамічних аспектів систем (рисунок 7.2). Діаграми варіантів використання – основний вид діаграм при моделюванні поведінки системи, підсистеми або класу. Кожна з них показує набір варіантів використання і діючих осіб в їх взаємодії.

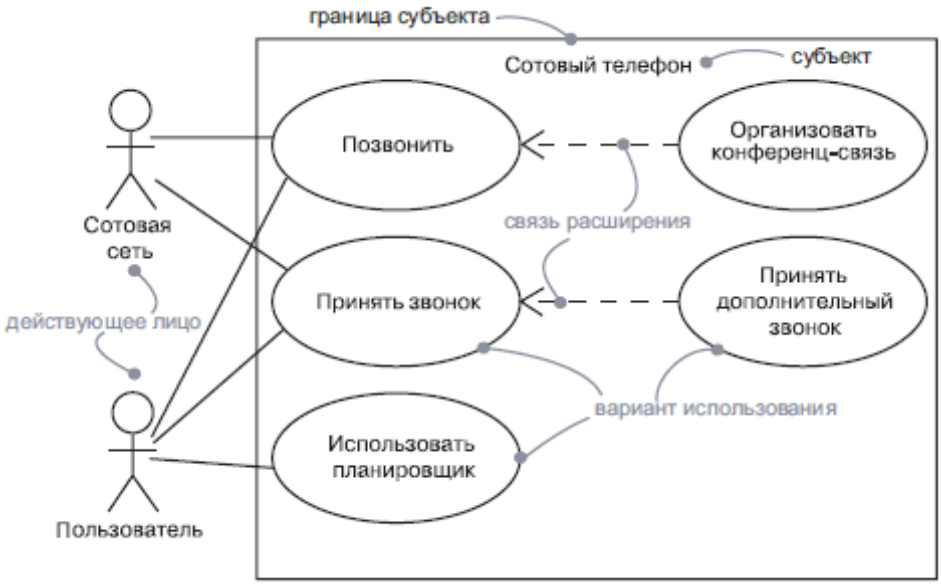
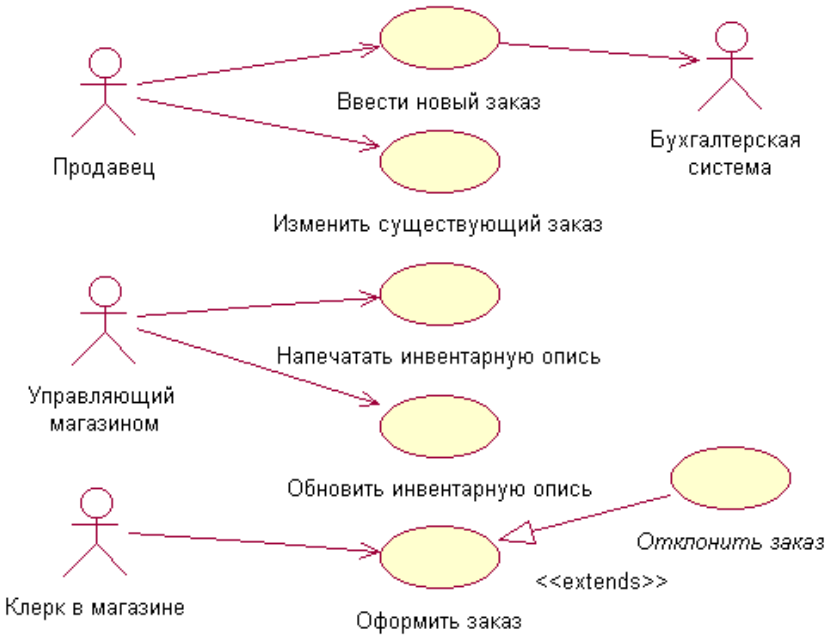


Рисунок 7.2 – Приклад діаграми варіантів використання

Розглянемо приклади діаграм варіантів використання.
Приклад 1.



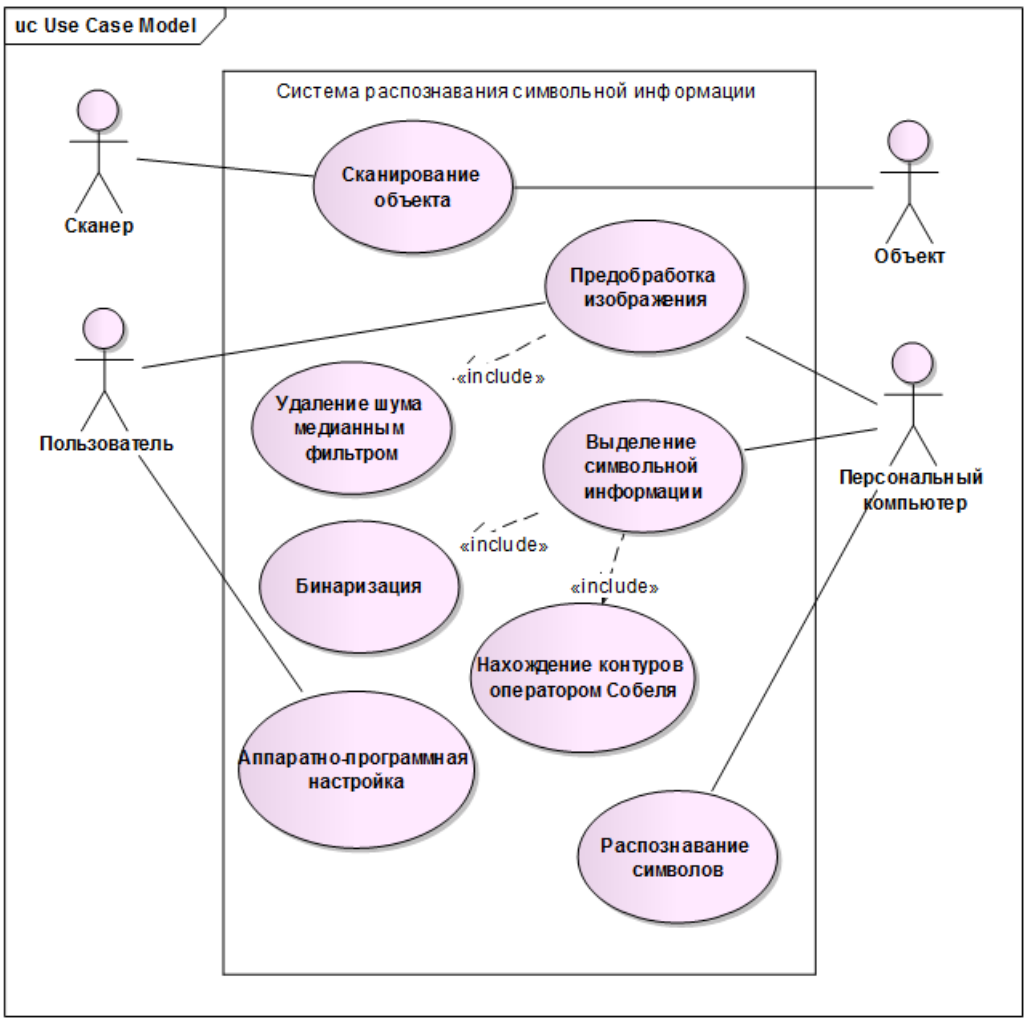
Приклад 2.



Приклад 3.



Приклад 4.



2. Діаграми станів

Діаграми станів – це один з п'яти видів діаграм UML, призначених для моделювання динамічних аспектів поведінки систем. Діаграми станів підходять для моделювання життєвого циклу об'єкта.

Однак в той час, як діаграма діяльності демонструє потік управління від однієї діяльності до іншої через безліч об'єктів, діаграма станів відображає потік управління від стану до стану всередині окремого об'єкта. Діаграми стану застосовуються для моделювання динамічних аспектів поведінки систем (в більшості випадків це моделювання поведінки реактивних об'єктів).

Реактивним називається такий об'єкт, поведінка якого найкраще характеризується його реакцією на події, одержувані ним ззовні його контексту.

Реактивний об'єкт має чіткий час життя, і його поточна поведінка залежить від минулого. Зазвичай реактивний об'єкт простоює в очікуванні події. Коли він отримує цю подію, його реакція зазвичай залежить від попередніх подій. Відреагувавши певним чином, об'єкт знову повертається в стан простою, чекаючи наступної події.

Діаграми станів можуть бути приєднані до класів, варіантів використання або до всієї системи з метою візуалізації, специфікування, конструювання та документування динаміки окремих об'єктів.

Діаграма станів (state diagram) показує автомат, зосереджуючи увагу на потоці управління від одного стану до іншого. Зображується у вигляді графа з вершинами та дугами (ребрами).

Автомат (state machine) – це опис послідовності станів, через які проходить об'єкт протягом життєвого циклу, реагуючи на події, а також опис реакції на ці події.

Стан (state) – ситуація в життєвому циклі об'єкта, протягом якої він задовольняє деякій умові, виконує деяку діяльність або очікує деяку подію.

Подія (event) – специфікація суттєвого факту, який відбувається в часі та просторі. В контексті автомата подія - це вплив, який викликає перехід між станами.

Перехід (transition) – зв'язок між двома станами, що показує, що об'єкт, що знаходиться в першому стані, повинен виконати деякі дії та перейти до другого, як тільки відбудеться певна подія і будуть виконані певні умови.

Діяльність (activity) специфікує роботу, яка відбувається всередині автомата.

Дія (action) – примітивне виконуване обчислення, що приводить до зміни стану моделі або поверненню значення.

Діаграми стану зазвичай містять прості та складові стани, а також переходи, події і дії. Крім того, як і всі діаграми, можуть включати примітки та обмеження.

Діаграми станів використовуються при моделюванні поведінки реактивних об'єктів, особливо екземплярів класів, варіантів використання та системи в цілому. У той час як взаємодії моделюють поведінку спільноти об'єктів, що працюють разом, діаграма станів моделює поведінку окремого об'єкта під час його життєвого циклу.

На відміну від діаграм діяльності, які моделюють потік управління від однієї діяльності до іншої, діаграма станів моделює потік управління від однієї події до іншої.

При моделюванні поведінки реактивного об'єкта, по суті, специфікуються три речі :

- стабільні стани, в яких об'єкт може перебувати,
- події, що викликають переходи від стану до стану,
- дії, що виконуються при кожній зміні станів.

Моделювання поведінки реактивного об'єкта має на увазі моделювання його життєвого циклу починаючи зі створення об'єкта аж до його знищення, з виділенням стабільних станів, в яких він може перебувати.

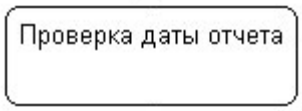
Стабільний стан представляє умову, при який об'єкт може існувати протягом деякого певного періоду часу. Коли відбувається подія, об'єкт може переходити з одного стану в інший.

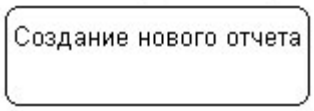
Ці події також можуть викликати переходи в себе і внутрішні переходи, коли поточна й цільовий стан об'єкта збігаються. Реагуючи на події та зміни стану, об'єкт може виконувати дії.

Щоб змоделювати реактивний об'єкт, необхідно:

- Вибрати контекст автомата – клас, варіант використання або систему в цілому
- Встановити початковий та кінцевий стани об'єкта.
- Прийняти рішення щодо стабільних станів об'єкта, розглянувши умови, в яких об'єкт може існувати протягом певного періоду часу.
- Прийняти рішення щодо подій, які можуть викликати переходи з одного стану в інший.
- Приєднати дії до переходів та / або до станів.
- Переконатися, що усі стани досяжні при деякому збігу обставин.
- Перевірити, чи немає яких-небудь «мертвих» станів, з яких об'єкт не зможе вийти ні при якому поєднанні подій.

Закруглені прямокутники представляють стани, через які проходить об'єкт протягом свого життєвого циклу.





Стан може містити тільки ім'я або ім'я та додатково список внутрішніх дій. *Список внутрішніх дій* містить перелік дій або діяльностей, які виконуються під час знаходження об'єкта в даному стані. Даний список фіксований.

Список основних дій включає наступні значення :

– entry – дія, яка виконується в момент входу в даний стан (вхідна дія);
– exit – дія, яка виконується в момент виходу з цього стану (вихідна дія);
– do – діяльність, яка виконується ("do activity") протягом усього часу, поки об'єкт знаходиться в даному стані
– defer – подія, обробка якої пропонується в іншому стані, але після того, як всі операції в поточному будуть завершені.

Наприклад:



Факт зміни одного стану іншим зображується за допомогою переходу. Перехід здійснюється при настанні деякої події:

– закінчення виконання діяльності (do activity),
– отриманні об'єктом повідомлення
– прийомом сигналу.

Перехід може бути тригерний та нетрігерним.

Якщо перехід спрацьовує, коли всі операції вихідного стану завершені, він називається *нетрігерним* або переходом по завершенні.

Якщо перехід ініціюється якою-небудь подією, він вважається *тригерним*. Для тригерного переходу характерна наявність імені, яке може бути записано в наступному форматі:

<ім'я події> (<список параметрів, розділених комами>) [<сторожова умова>] <вираз дії>

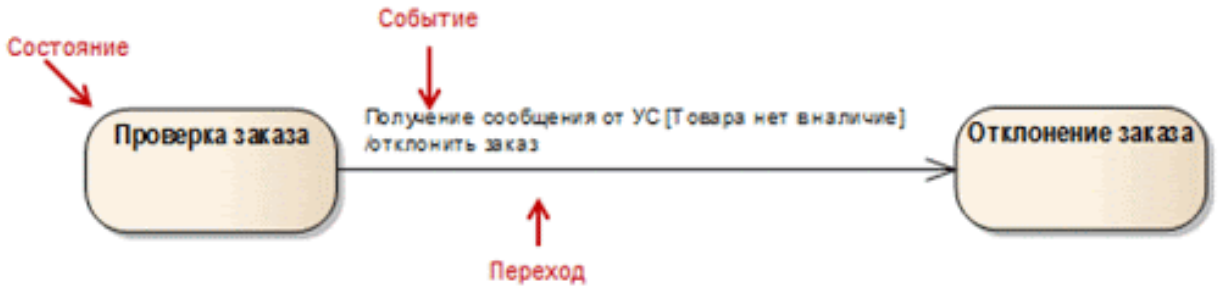
В якості події можуть виступати сигнали, виклики, закінчення фіксованих проміжків часу або моменти закінчення виконання певних дій.

Сторожеве умова (guard condition) завжди записується в прямих дужках після події-тригера і являє собою деякий булевский вираз. У загальному випадку з одного стану може бути кілька переходів з однією і той же подією-тригером, при цьому цільовий стан буде залежати від того яке з сторожових умов прийме значення «істина».

Також ім'я переходу може містити *вислів дії* (action expression). В даному випадку вказану дію виконується відразу при спрацьовуванні переходу і до початку будь-яких дій в

цільовому стані. У загальному випадку вислів дії може містити цілий список окремих дій, розділених символом «;».

Стрілками показуються переходи між станами, які викликані виконанням подій описуваного діаграмою об'єкта.

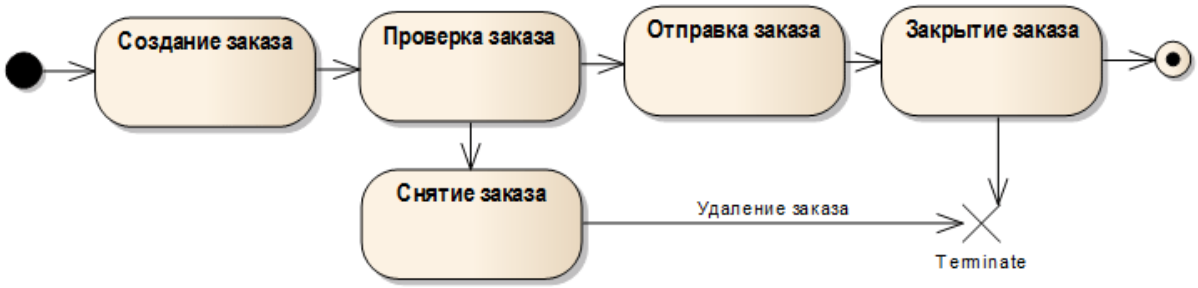


Існує також два види псевдостанів:

– початкове, в якому знаходиться об'єкт відразу після його створення (позначається суцільним кружком),
– кінцеве, яке об'єкт не може покинути, якщо перейшов в нього (позначається кружком, обведеним окружністю).

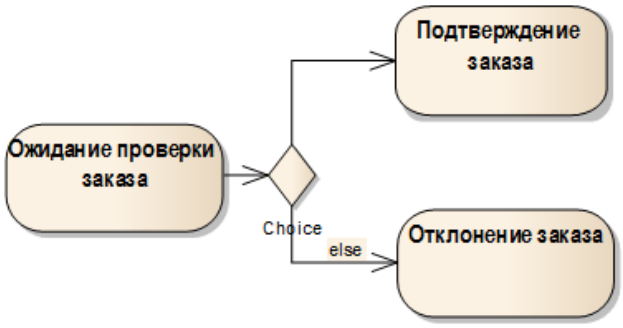
Якщо необхідно відобразити знищення об'єкта використовується вузол завершення (terminate node) – псевдостан, вхід в який означає завершення виконання поведінки кінцевого автомата в контексті його об'єкта.

Наприклад:



Варіанти прийняття рішень на діаграмі станів можуть бути показані також як і на діаграмі діяльності за допомогою *вузла вибору*. При цьому перехід в стан вибору повинен бути тригерним і містити ім'я події. Переходи з псевдостану вибору в цільові стани повинні містити сторожові умови. Перехід, який повинен спрацьовувати, якщо жодна з умов не прийме значення «істина» повинен містити позначку «else».

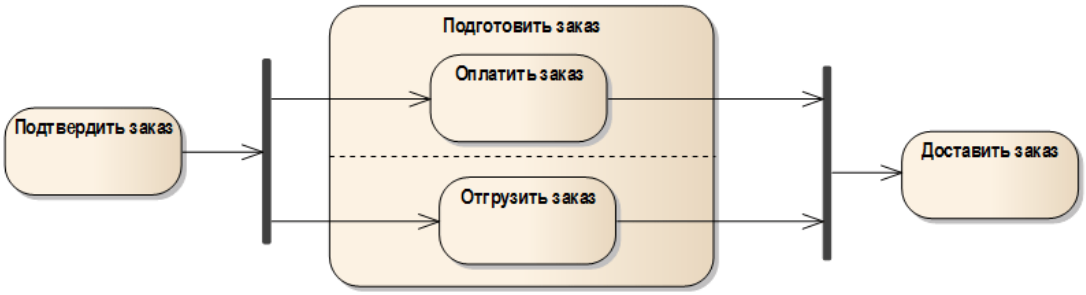
Наприклад:



На діаграмі можуть бути представлені як прості стани, так і складні стани. *Складні* або *складові стани* (composite state) включають в себе вкладені підстани.

Вузли розгалуження та об'єднання аналогічні вузлам на діаграмі діяльності. Основна мета даних підстанів показати паралельну роботу підавтомат. Після спрацьовування переходу модельований об'єкт одночасно буде перебувати у всіх цільових станах цього переходу.

Наприклад:



Переходи можуть здійснюватися як в композитний стан, так і в одне з його підстанів. Таким чином, перехід, стрілка якого з'єднана з кордоном деякого складного стану, позначає перехід до складного стан. Він еквівалентний переходу в початковий стан кожного з підавтомат.

Перехід, що виходить з складного стану відноситься до кожного з вкладених підстанів. Це означає, що об'єкт може покинути складовий суперстан, перебуваючи в будь-якому з його підстанів. Якщо необхідно вказати конкретний підстан з якого може здійснитися вихід з композитного стану, досить додати перехід від підстану в цільовий стан.

Наприклад:



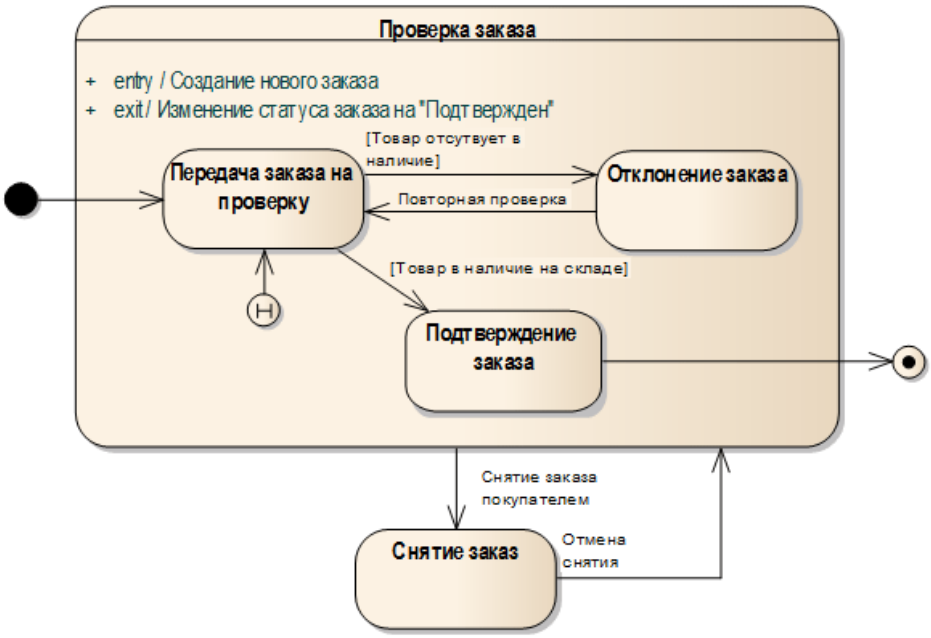
Підстани можуть бути:

– послідовними (неортогональні)
– паралельними (ортогональні)

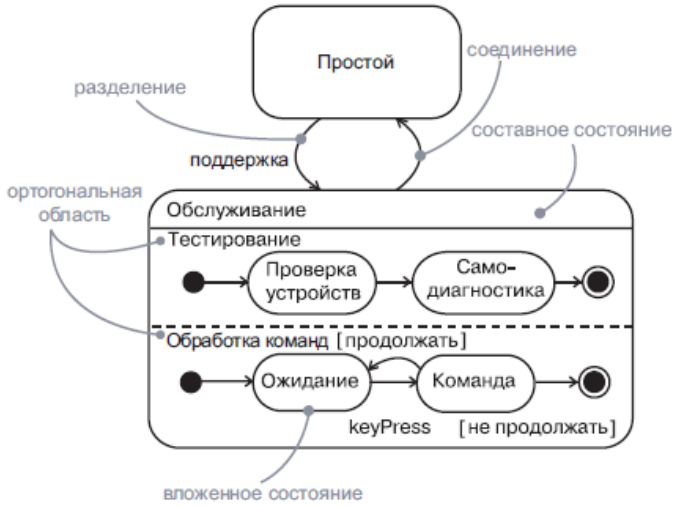
Послідовні підстани (неортогональні) використовуються для моделювання такої поведінки об'єкта, під час якої в кожен момент часу об'єкт може знаходитися в одному і тільки одному підстані. Поведінка об'єкту в цьому випадку представляє собою послідовну зміну підстанів, починаючи від початкового і закінчуючи кінцевим підстаном.

Паралельні підстани (ортогональні) дозволяють специфікувати два і більше підавтоматів, які можуть виконуватися паралельно всередині складної події. Кожен з підавтоматів займає деяку область (регіон) всередині складного стану.

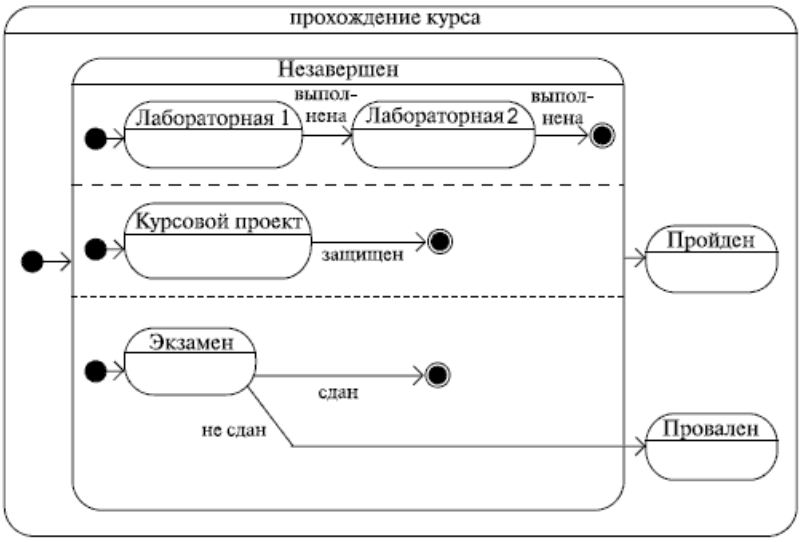
Наприклад:



Історичний стан (history state) дозволяє складовому стану, що включає неортогональні підстани, «пам'ятати» підстан, який був активним на момент останнього переходу з складного стану зовні. Історичний стан зображується у вигляді маленького кружечка з буквою «H» (History).



Приклад діаграми станів:



Запитання для самоперевірки

1. Що таке варіант використання?
2. Що таке діюча особа?
3. Які зв'язки можна організувати між варіанти використання?
4. Для чого використовується діаграма станів?
5. Розкрийте поняття стан, подія, перехід та діяльність.

Лекція 8. НОТАЦІЯ BPMN

1. Ролі або зони відповідальності
2. Об'єкти потоку управління

1. Ролі або зони відповідальності

BPMN (Business Process Management Notation) – це мова моделювання бізнес-процесів, який є проміжною ланкою між формалізацією / візуалізацією і втіленням бізнес-процесу.

Нотація BPMN є опис графічних елементів, які використовуються для побудови схеми протікання бізнес-процесу. Моделювання BPMN дозволяє згодом провести автоматизацію бізнес-процесів відповідно до наявної схеми.

BPMN-процес – це будь-який бізнес-процес, відбитий за допомогою нотації. Процеси складаються з елементів, кожен з яких позначається на схемі спеціальним значком.

Нотація спирається на наступні базові графічні елементи:

– Ролі або зони відповідальності (Swimlanes): пул та доріжки.
– Об'єкти потоку управління (Flow Objects): події, дії та логічні оператори.
– З'єднуючі об'єкти (Connecting Objects): потік управління, потік повідомлень та асоціації.
– Артефакти (Artifacts): дані, групи та текстові анотації.

Весь бізнес-процес складається з *Пулів* (Pool) – це сукупності операцій та осіб, які ці операції виконують.

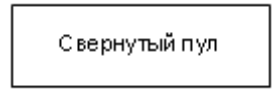
Пул являє собою:

– учасника взаємодії.
– може виступати в якості Зони відповідальності або графічного контейнера, що відповідає за розподіл певного набору дій, що відносяться до інших Пулів

Також пул використовується для позначення меж бізнес-процесу. Позначається пул наступним чином:



Згорнутий пул – елемент, що позначає зовнішній (по відношенню до поточної діаграмі) процес або зовнішнє посилання. Усередині блоку поміщається найменування зовнішнього процесу або зовнішнього посилання.



Згорнутий пул використовується для вказівки взаємозв'язків процесу:

– позначає процес або зовнішнє посилання, звідки надійшов або куди передається потік повідомлень;
– позначає попередній або наступний процес по відношенню до діаграми даного процесу.

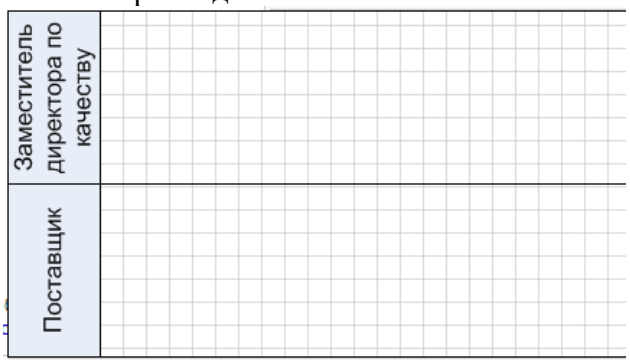
Наприклад:



Доріжка (Lane) використовується для відображення розподілу ролей і може бути як вертикальної, так і горизонтальної (також може використовуватися для розділення внутрішнього простору Пула). Служить для упорядкування та категоризації Дій. Позначається:



Наприклад:



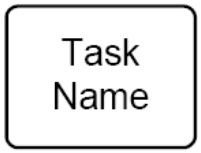
2. Об'єкти потоку управління

У кожній доріжці розташовуються дії, що виконуються одним виконавцем.

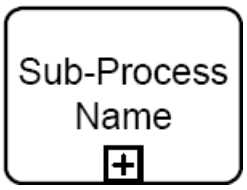
Під *Дією* розуміється одиниця роботи, що виконується в ході виконання бізнес-процесу. Дії можуть бути як елементарними (завдання / task), так і складовими (підпроцес / sub-process).

BPMN передбачає наступні графічні відображення для основних типів дій:

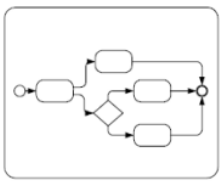
Абстрактна задача – використовується для позначення простого дії або операції, яка не має подальшої декомпозиції в рамках поточного бізнес-процесу.



Підпроцес – використовується для відображення декомпозированого процесу, включеного до складу даного процесу. Діаграма не відображує деталі підпроцесу.



Згорнутий



Розгорнутий

Процес-посилання – використовується для позначення посилання на один з найбільш часто повторюваних процесів.



Користувацьке завдання – використовується для відображення завдання, яку виконує людина



Завдання на виконання сценарію – використовується для відображення кроку процесу, після досягнення якого автоматично виконується скрипт.



Завдання на виклик сервісу – використовується для ілюстрації кроку процесу, на якому викликається веб-служба або скрипт C#.



Вбудований кейс – використовується для представлення нестандартної задачі, яку курує відповідальна особа або група осіб. Кейси використовуються, коли потрібно швидко організувати в рамках процесу неструктуровану або слабоструктуровану активність.



Подія є одним з головних елементів BPMN і служить для опису того, що має статися (на відміну від завдання, коли щось має бути зроблено). Подією може бути, наприклад, підписання договору, або розмова з клієнтом.

Графічні елементи подій в BPMN класифікують двома способами:

– Залежно від положення події на схемі процесу;
– За типом події.

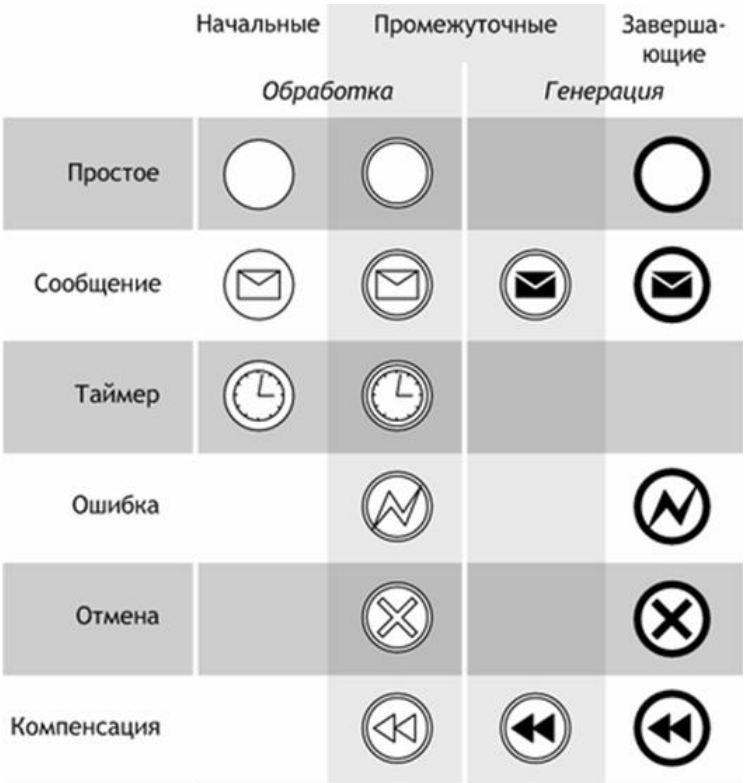
Залежно від положення події на схемі процесу :

– Початкова подія (ініціює бізнес-процес) 

– Проміжна подія 

– Кінцеве подія (що закінчує бізнес-процес) 

За типом події класифікація продемонстрована в таблиці:





Прості події (plain events) це нетипізовані події, які використовуються, найчастіше, для того, щоб показати початок або закінчення процесу.

Події-повідомлення (message events) показують отримання і відправку повідомлень в ході виконання процесу.



Події-таймери (timer events) моделюють події, регулярно відбуваються у часі. Також дозволяють моделювати моменти часу, періоди і тайм-аути.



Події-помилки (error events) дозволяють змоделювати генерацію і обробку помилок в процесі. Помилки можуть мати різні типи.



Події-скасування (cancel events) ініціюють або реагують на скасування транзакції.



Події-компенсації (compensation events) ініціюють компенсацію або виконують дії по компенсації.



Події-умови (conditional events) дозволяють інтегрувати бізнес правила у процес.



Події-сигнали (signal events) розсилають і приймають сигнали між декількома процесами. Один сигнал може оброблятися декількома одержувачами. Таким чином, події-сигнали дозволяють реалізувати трансляцію розсилання повідомлень.



Події нижчого рівня (multiple events) моделює генерацію і моделювання однієї події з безлічі.



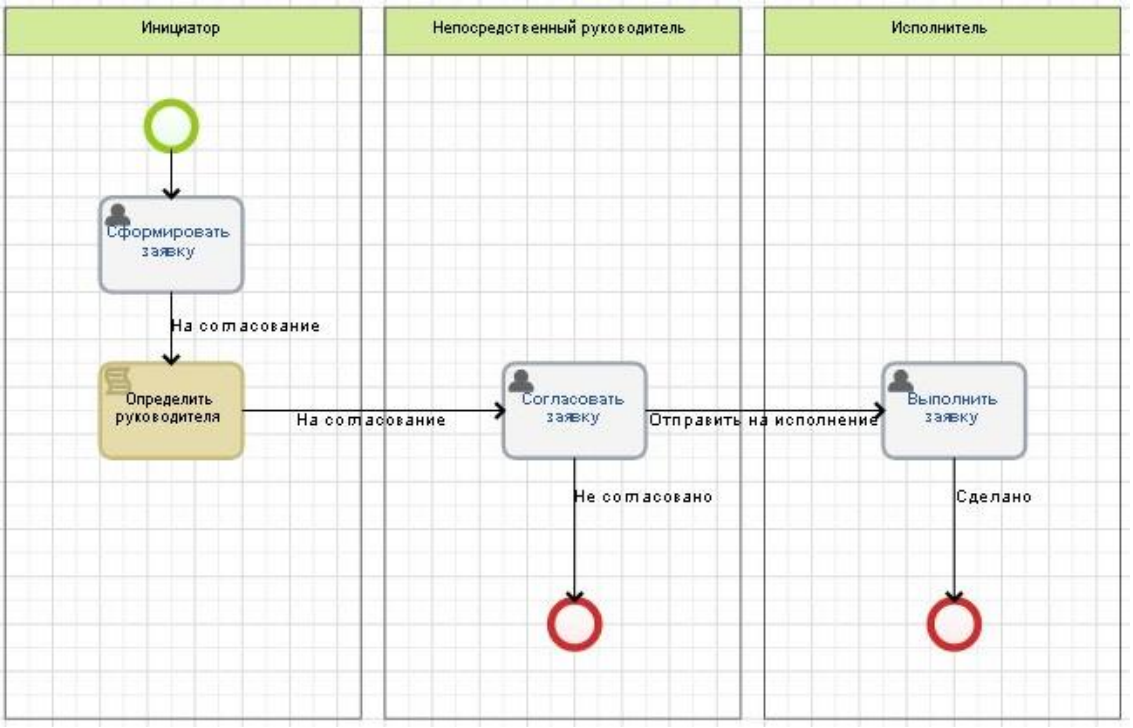
Події-посилання (link events) використовуються як міжсторінкових з'єднання. Пара відповідних посилань еквівалентна потоку управління.



Події-останови (terminate events) призводять до негайного завершення всього бізнес процесу (у всій діаграмі).

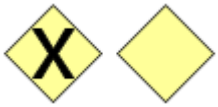


Наприклад, продаж та відвантаження обладнання:

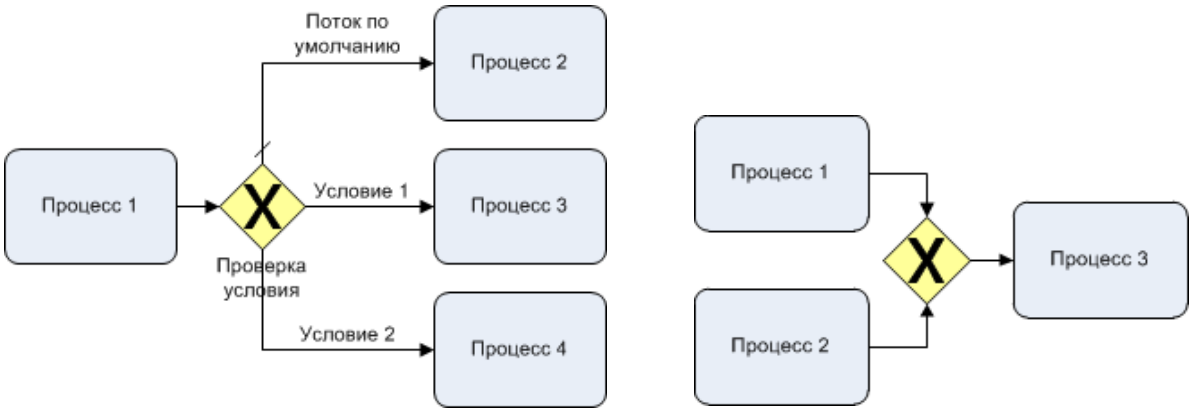


Під *шлюзами* розуміються елементи, що визначають розгалуження та злиття потоків робіт. Шлюз можна уявити як пропускний пристрій, який або пропускає потік, або ні.

Шлюз виключне «або» – використовується для створення альтернативних потоків процесу або що сходяться потоків управління. Оцінює стан бізнес-процесу і, в залежності від умови, розбиває потік на одне або кілька взаємовиключних напрямів.



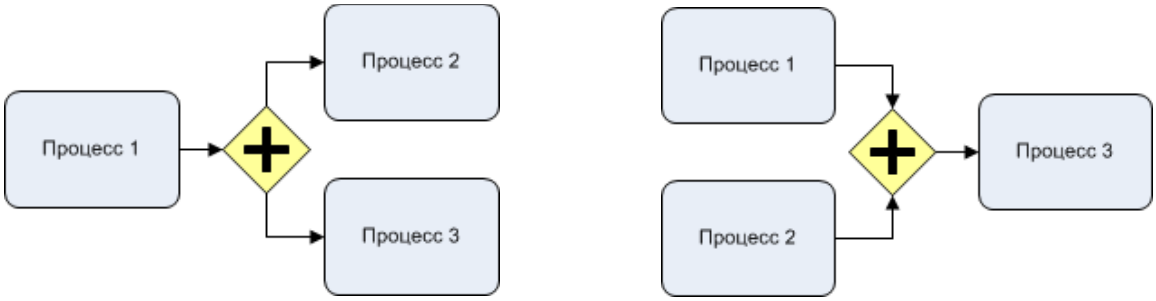
Наприклад:



Паралельний шлюз – використовується для створення паралельних шляхів без оцінки якої б то не було умови або для потоків, що сходяться та синхронізації паралельних гілок виконання процесу.

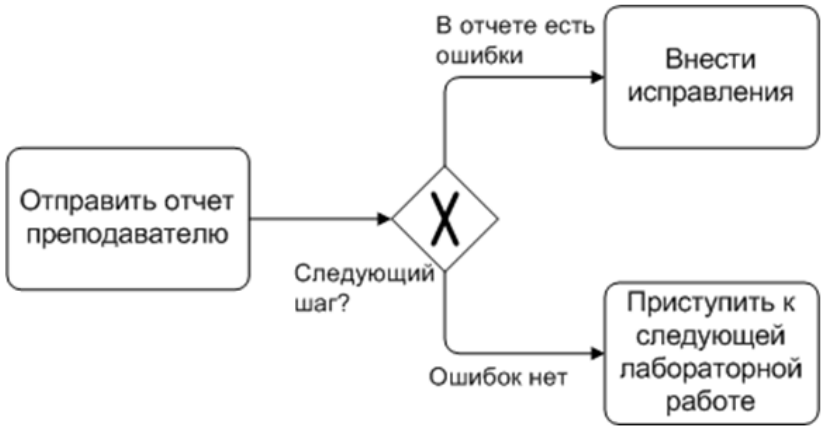


Наприклад:

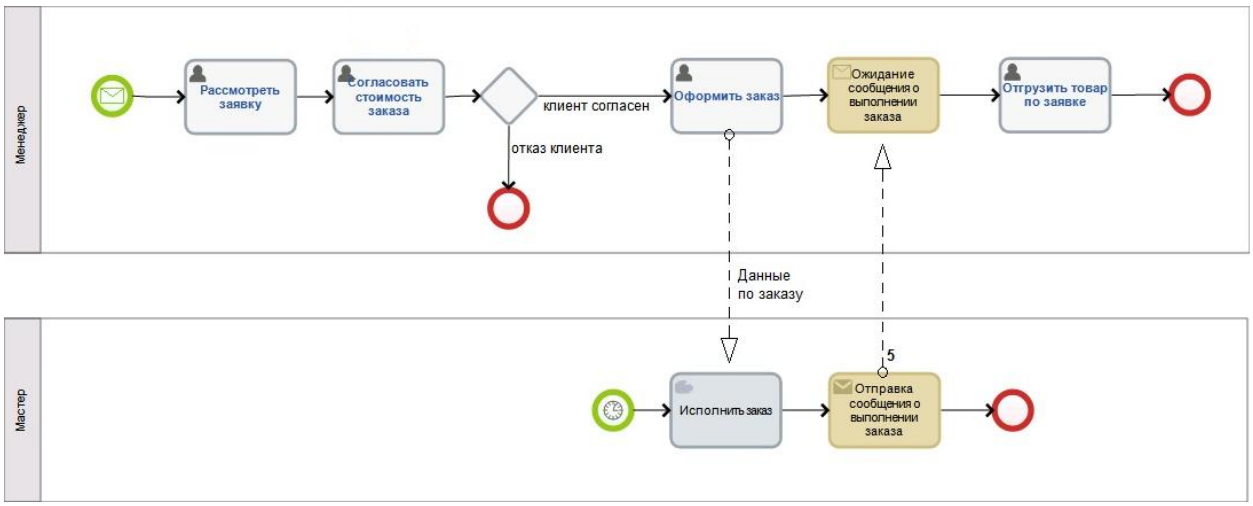


Двох розвилок, описаних вище досить для побудови бізнес-процесів будь-якої складності. Решта типів розвилок, описаних в BPMN, дозволяють будувати більш компактні схеми процесів.

Приклад:



Приклад:

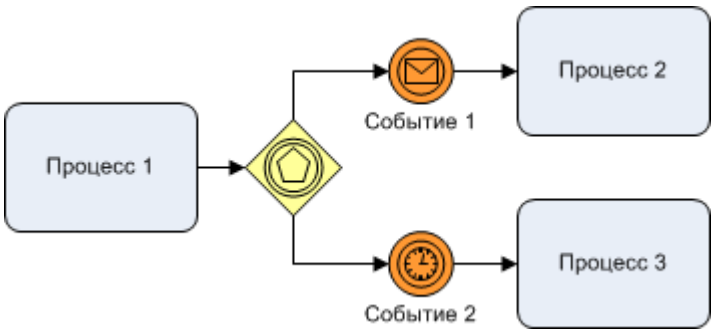


Залежність від події – розвилка з залежністю від події схожа на розвилку «виключення». Однак в разі залежності від події напрямок диктується тим, яка була вчинена подія, а не тим, яка була виконана умова.

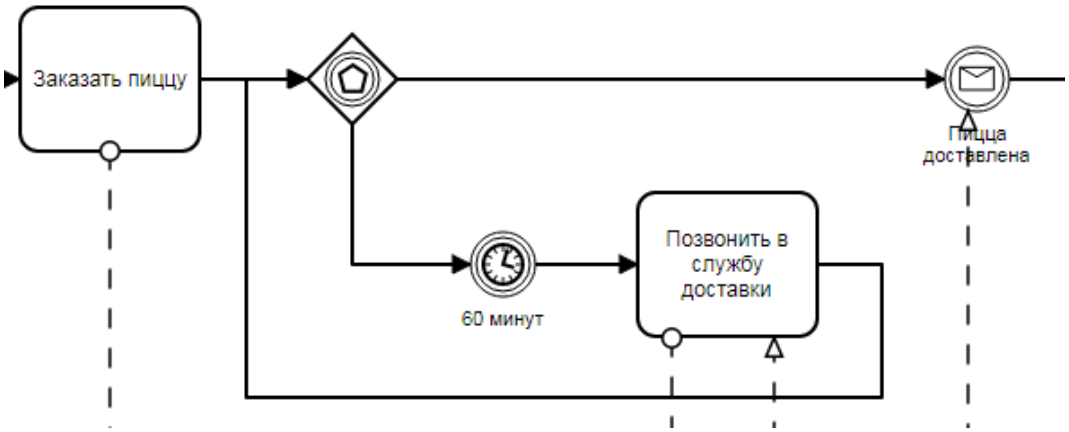


Припустимо, перш ніж відправити електронний лист, ви вирішили дочекатися приходу гендиректора в офіс. Якщо ж він не з'явиться, лист не буде відправлено.

Наприклад, якщо першим виникла Подія 1, то виконається тільки Процес 2; якщо першим виникла Подія 2, то виконається тільки Процес 3.



Наприклад. Після дії «Замовити піцу» розташований ексклюзивний шлюз по подіям, що показує, що далі процес піде по тій гілці, подія по якій настане раніше.



Можливо два варіанти:

– піца доставлена, як зазначено в проміжній події з типом «Отримання повідомлення» - «Піца доставлена»,

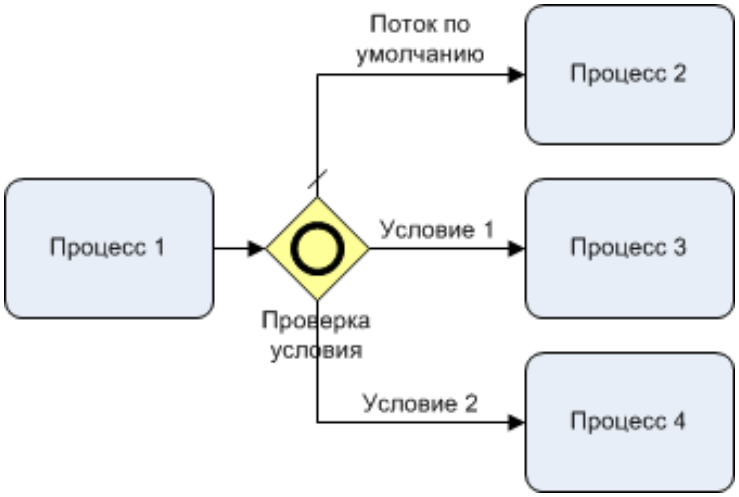
– піца не була доставлена протягом 60 хвилин (подія «Таймер»), в цьому випадку клієнт дзвонить в службу доставки.

Включення – розбиває потік процесу на одне або кілька напрямків. Наприклад, розвилка «включення» може вказувати на дії, вжиті компанією в результаті отриманих результатів опитування.



Скажімо, якщо покупцеві сподобався товар А, це спровокує один процес. Якщо покупець віддав перевагу товару Б, в силу вступить інший процес. А якщо покупець залишився незадоволений товаром А, – третій.

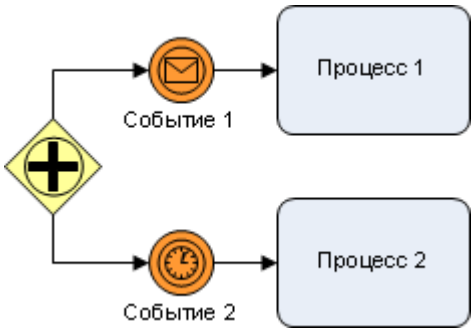
Наприклад, якщо Умова 1 вірно, то виконається Процес 3; якщо Умова 2 вірно, то виконається Процес 4; якщо ні Умова 1, ні Умови 2 неправильні, то виконається тільки Процес 2.



Паралельна розвилка з залежністю від події – схожа на паралельну розвилку, як видно з назви. Дозволяє декільком процесам розгортатися одночасно, тільки на відміну від паралельної розвилки, процеси залежать від конкретних подій.



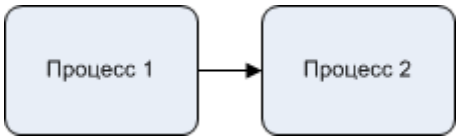
Приклад позначення:



Потік – це послідовність дій, яка позначається стрілкою. Елемент «потік» показує яку дію після якого необхідно здійснити.

Потік управління – на стандартний потік керування не впливають умови і він не проходить через шлюзи, тобто є неконтрольованим.





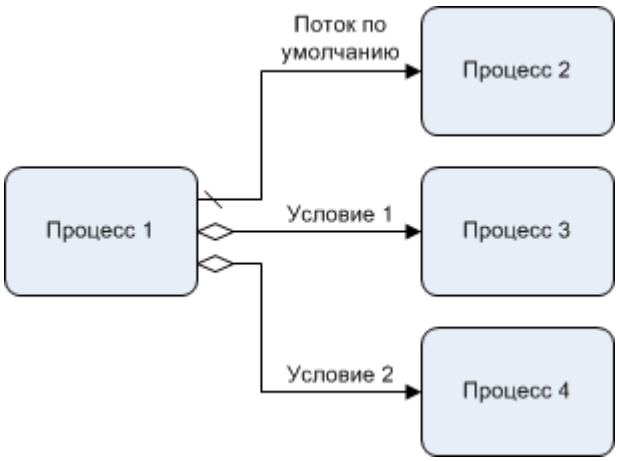
Потік управління за замовчуванням – використовується тоді, коли необхідно показати, що подальше виконання процесу буде відбуватися за певним потоком тільки якщо не виконується жодна з заданих умов.



Умовний потік управління – використовується для того, щоб показати, що подальше виконання процесу буде відбуватися за певним потоком тільки в тому випадку, якщо буде виконано задану умову.



Приклад позначення:

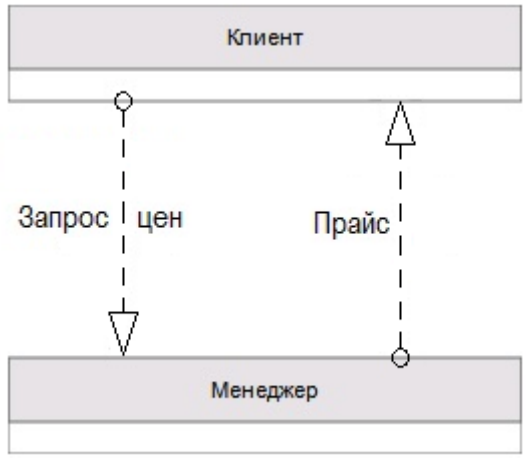


Ромбик біля основи стрілки додається, якщо умовний потік управління є вихідним від процесу. Ромбик не додають, якщо умовний потік управління є вихідним від шлюзу.

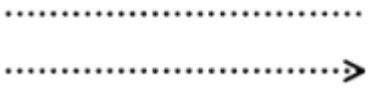
Потік повідомлень – використовується для відображення взаємодії між процесами - відображає передачу повідомлень або об'єктів з одного процесу в інший процес або зовнішнє посилання.

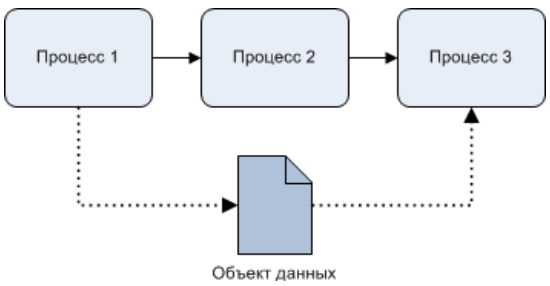


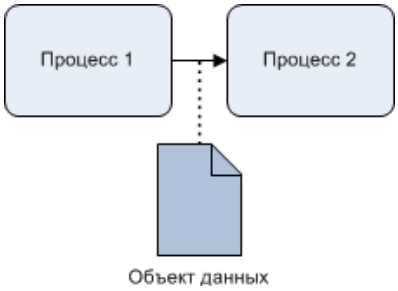
Наприклад, потік повідомлень між учасниками процесу:



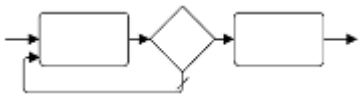
Асоціація – застосовується для візуалізації зв'язку між елементами потоку і об'єктами, які не є елементами потоку (артефактами).



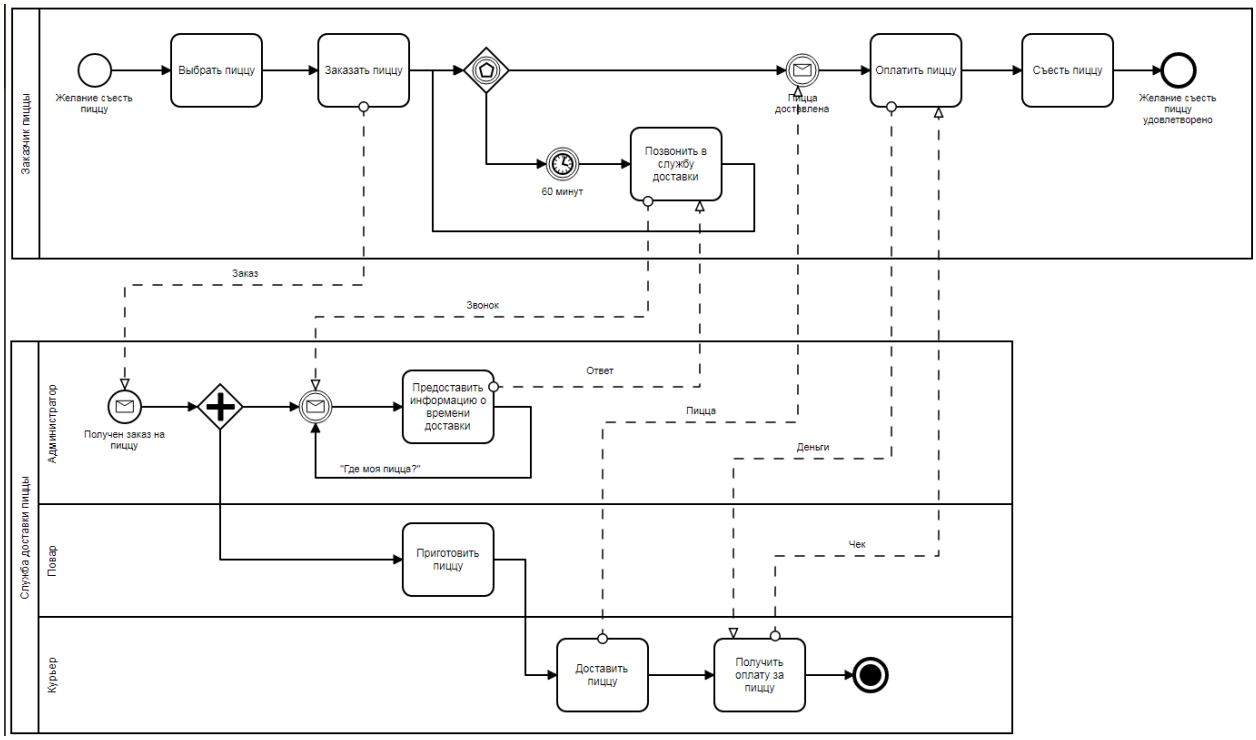




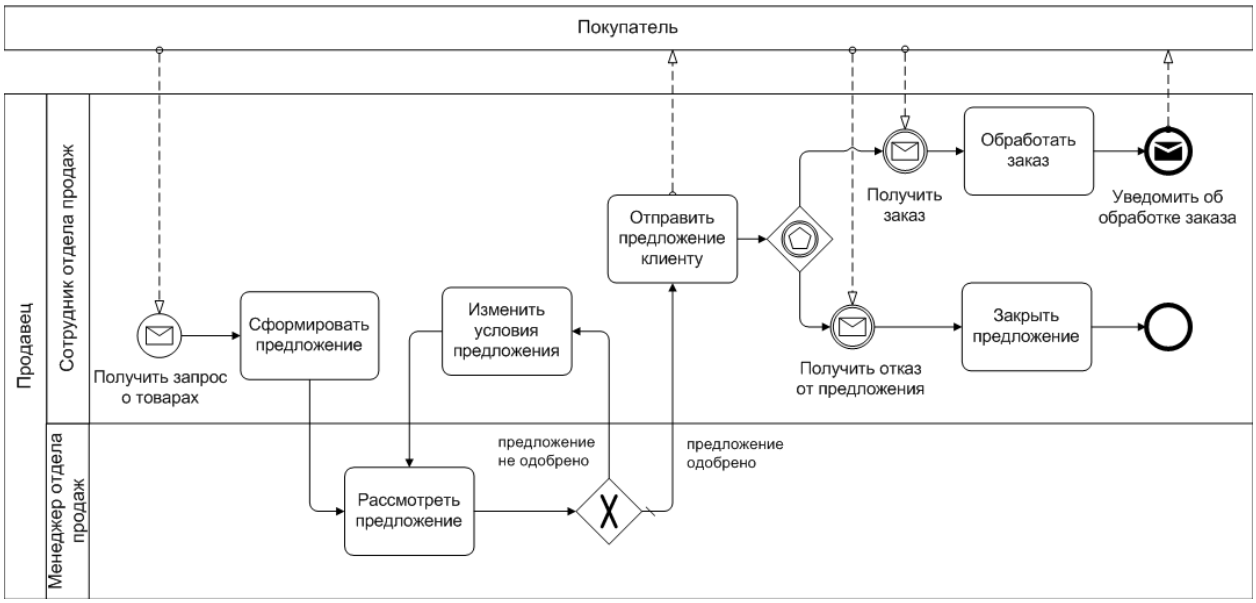
Цикли можуть з'являтися завдяки приєднанню Потоку операцій до «протилежного» об'єкту.



Приклад циклу:



Приклад циклу:



Під *артефактами* в BPMN розуміють об'єкти, які не впливають на виконання бізнес-процесу безпосередньо. Це можуть бути документи, дані, інформація.

Існує три різновиди артефактів – анотації, групи і дані. Всі три доповнюють або описують BPMN-процес.

– *Анотації* дозволяють модератору описати додаткові ділянки потоку моделі або нотації.



– *Групи* допомагають об'єднувати завдання або процеси, які важливі в контексті загального процесу.



– Дані – інформація, вміщена в процес, що виникла в результаті процесу або вимагає збору або зберігання



Введення даних – потреба в певній інформації, від якої залежить виконання завдань в бізнес-процесі.



Виведення даних – інформація, отримана в результати виконання бізнес-процесу.



Збір даних – інформація, зібрана в межах бізнес-процесу.

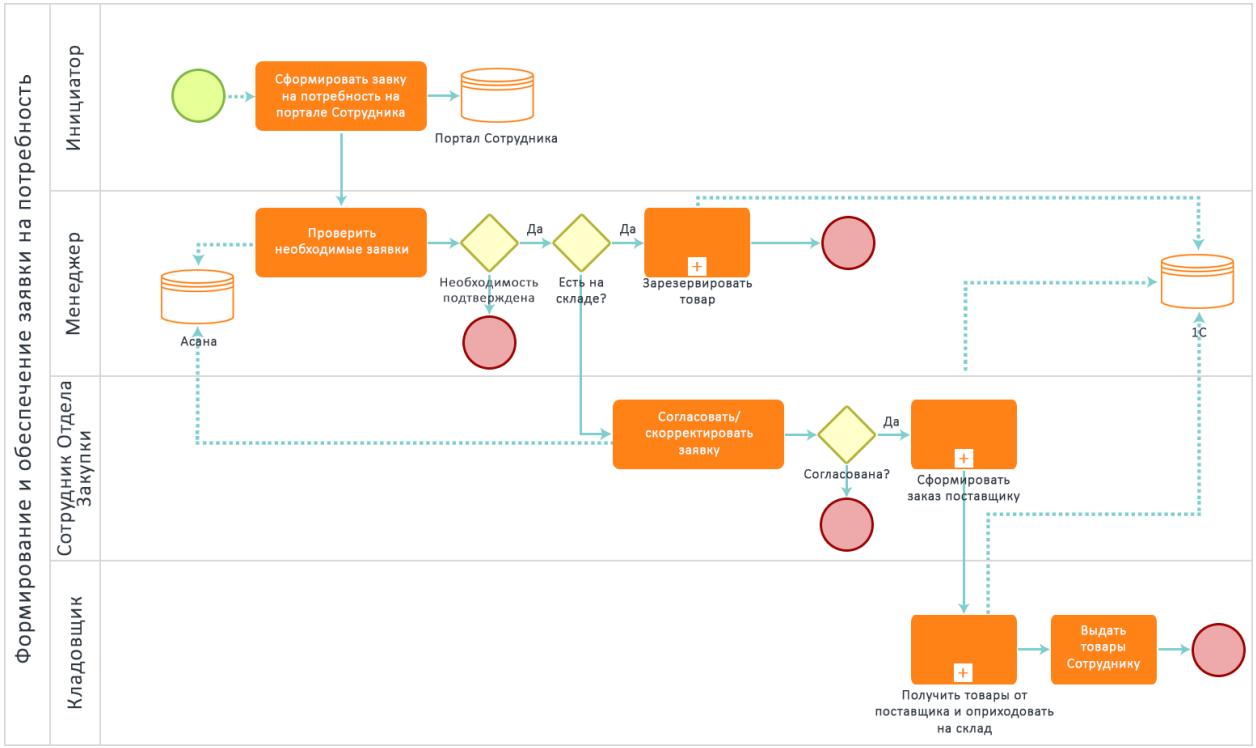


Сховище даних – можливість зберігати інформацію або отримати доступ до даних, пов'язаних з бізнес-процесом.

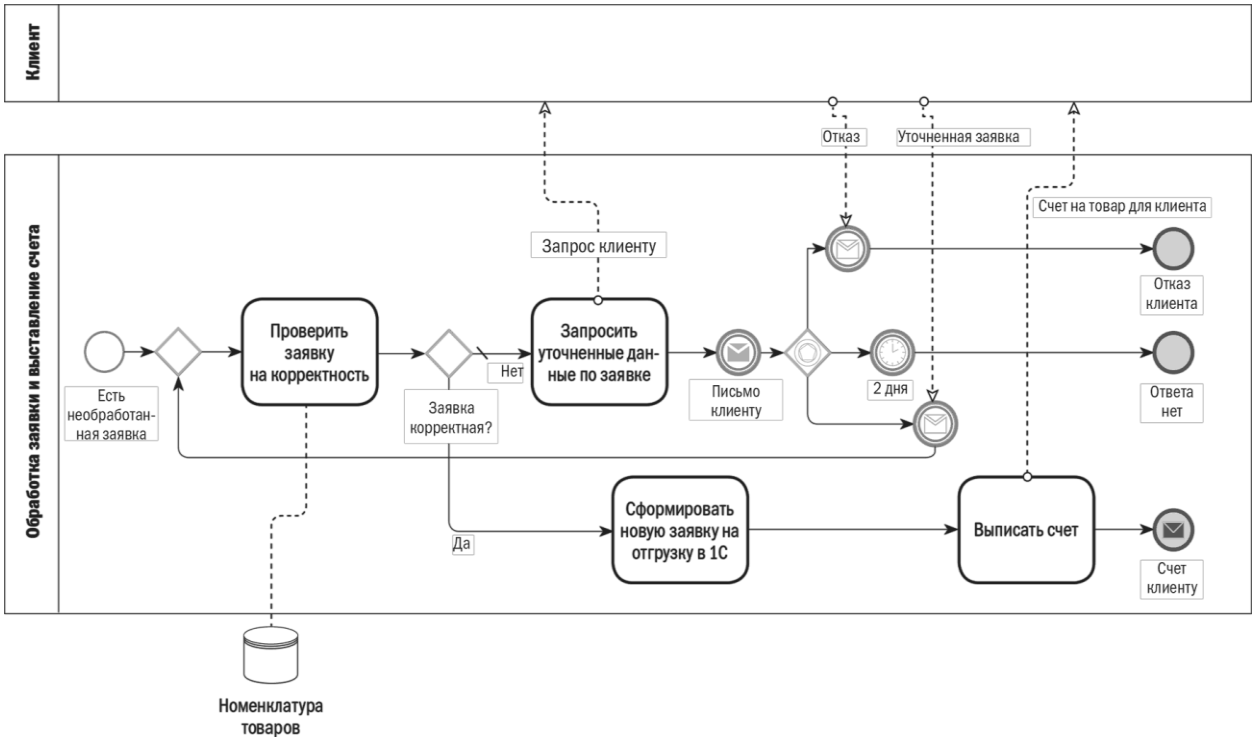


Розглянемо приклади опису задач за допомогою нотації BPMN.

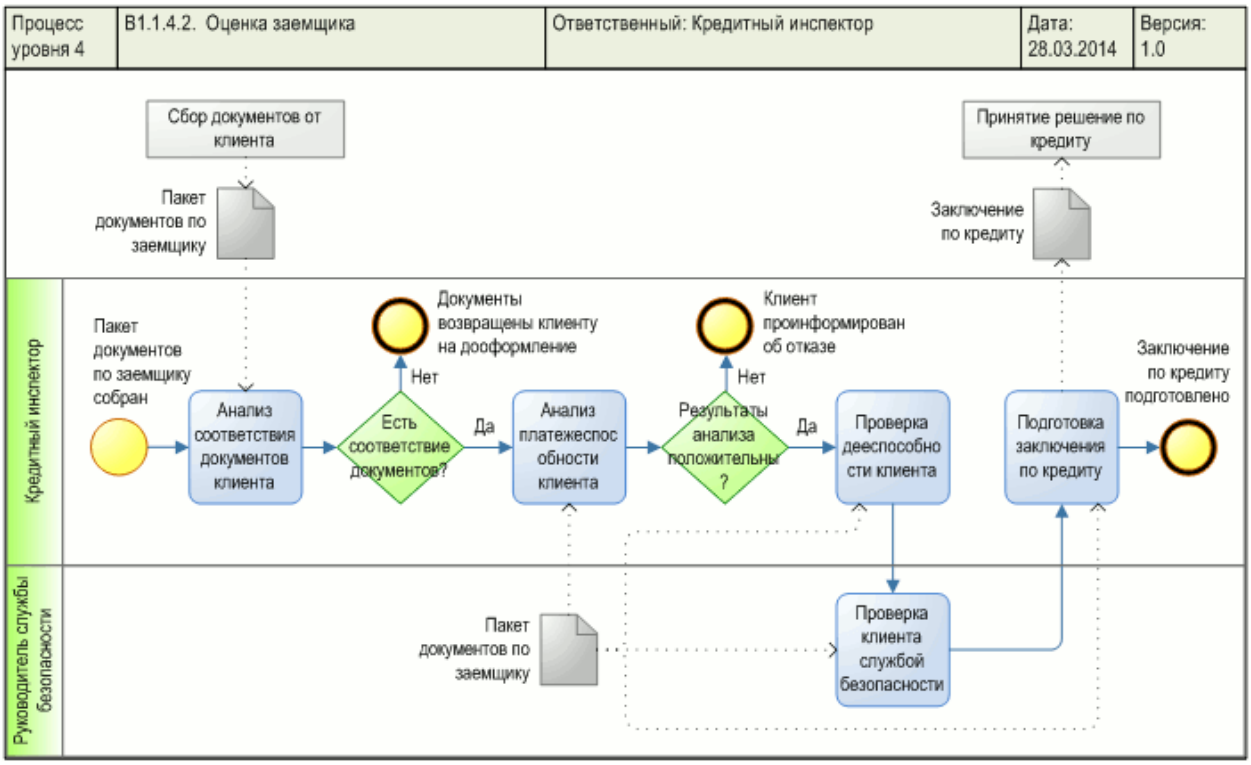
Приклад 1:



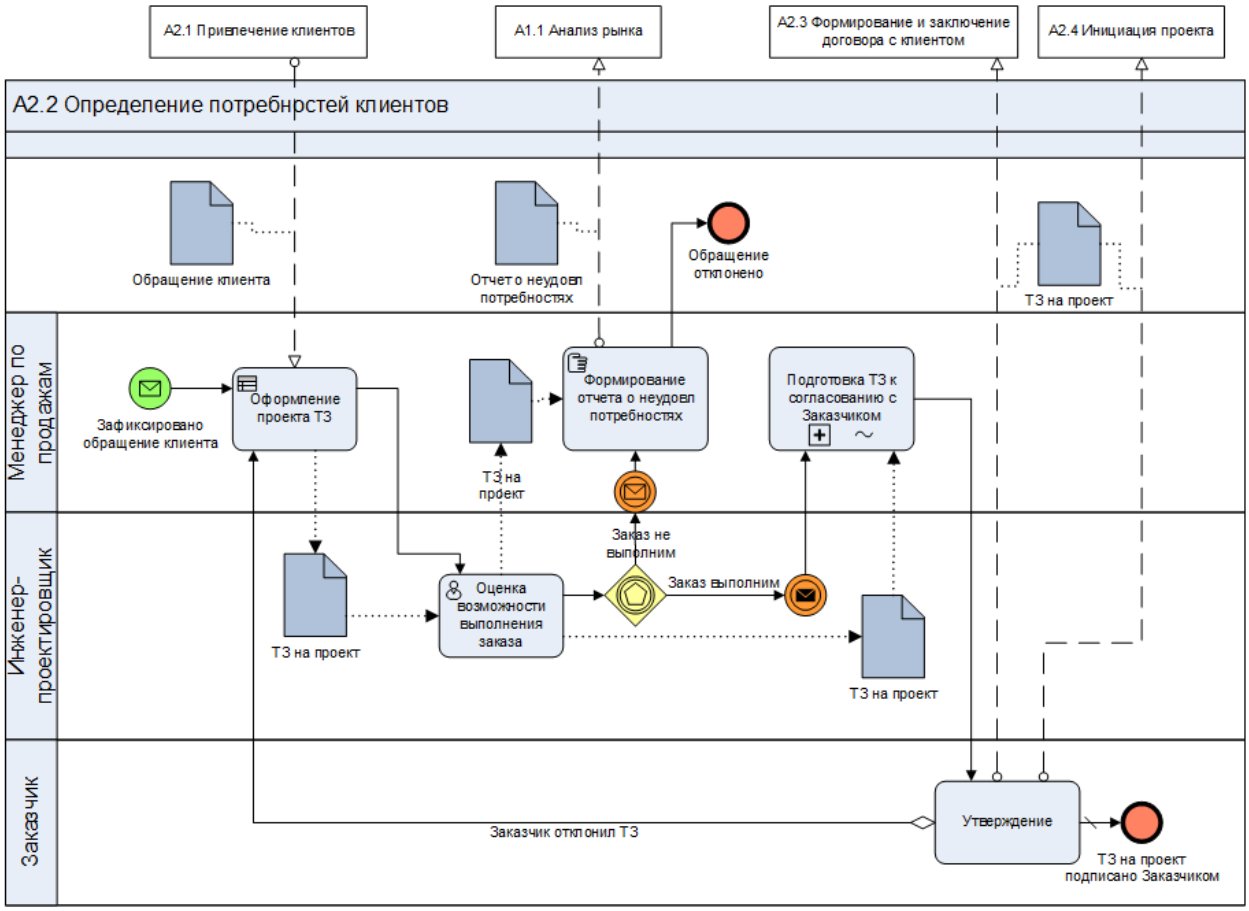
Приклад 2.



Приклад 3.


Процесс уровня 4
B1.1.4.2. Оценка заемщика
Ответственный: Кредитный инспектор
Дата: 28.03.2014
Версия: 1.0
Сбор документов от клиента
Пакет документов по заемщику
Принятие решение по кредиту
Заключение по кредиту
Кредитный инспектор
Руководитель службы безопасности
Пакет документов по заемщику собран
Анализ соответствия документов клиента
Есть соответствие документов?
Нет
Документы возвращены клиенту на дооформление
Да
Анализ платежеспособности клиента
Результаты анализа положительны?
Нет
Клиент проинформирован об отказе
Да
Проверка дееспособности клиента
Подготовка заключения по кредиту
Заключение по кредиту подготовлено
Пакет документов по заемщику
Проверка клиента службой безопасности

Приклад 4.


A2.1 Привлечение клиентов
A1.1 Анализ рынка
A2.3 Формирование и заключение договора с клиентом
A2.4 Инициация проекта
A2.2 Определение потребнрстей клиентов
Обращение клиента
Отчет о неудовл потребностях
Обращение отклонено
ТЗ на проект
Менеджер по продажам
Инженер-проектировщик
Заказчик
Зафиксировано обращение клиента
Оформление проекта ТЗ
ТЗ на проект
Формирование отчета о неудовл потребностях
Подготовка ТЗ к согласованию с Заказчиком
Заказ не выполним
Заказ выполним
ТЗ на проект
Оценка возможности выполнения заказа
ТЗ на проект
Заказчик отклонил ТЗ
Утверждение
ТЗ на проект подписано Заказчиком

Запитання для самоперевірки

1. В яких ситуаціях використовується нотація BPMN?
2. Поясніть поняття доріжка та пул. Для чого вони використовуються?
3. Яка основні типи дій в нотації BPMN ви знаєте?
4. Як класифікуються графічні елементи подій в BPMN? Для чого вони використовуються?
5. Поясніть поняття потік. Для чого він використовуються?
6. В яких кроків складається моделювання завдання в нотація BPMN?
7. Як позначається зв’язок з завданнями, які мають свою діаграму в нотація BPMN?
8. Як використовуються артефакти при моделюванні завдання в нотація BPMN?

Лекція 9. Нотація DFD

1. Компоненти синтаксису DFD
2. Варіанти графічних нотацій
3. Приклади використання DFD

1. Компоненти синтаксису DFD

Діаграма потоків даних або DFD (Data Flow Diagram) – це методологія графічного структурного аналізу, що описує зовнішні стосовно системи джерела та адресати даних, логічні функції, потоки даних та сховища даних, до яких здійснюється доступ.

Методологію DFD по праву вважається одним з основних інструментів структурного аналізу та проектування інформаційних систем, що існувала до поширення та застосування уніфікованої мови моделювання створення абстрактних моделей систем UML.

Правильно побудована діаграма у методології DFD дасть відповіді на такі питання як:

- Яка структура проектованої інформаційної системи?
- Як інформація та дані взаємодіють із системою та циркулюють усередині проектованої системи?
- Що необхідно, щоб потоки інформації в системі, що моделюється, були оброблені?

Діаграми потоків даних відомі дуже давно і були запропоновані Ларі Костянтином у 70-ті роки. XX ст. Однак, є ще більш рання їхня згадка, що відноситься до 1920-х років. Так, у літературі згадується можливе використання діаграми для оптимізації простору в офісі для роботи клерків. Під час здійснення реорганізації фахівець позначив гуртком кожного клерка, а стрілкою – кожен документ, що передається між ними. Намалювавши таким чином діаграму, він запропонував схему оптимізації, відповідно до якої клерки, що здійснюють максимальну передачу документів між собою, були посаджені поряд, а клерки з невеликою взаємодією на великій відстані.

Основна мета побудови діаграми у даній методології: візуалізація процесу передачі об'єктів (інформації) між учасниками цього процесу, а саме – демонстрація того, як кожен процес перетворює свої вхідні дані у вихідні та виявлення відносин між цими процесами. Діаграма наочно демонструє шляхи, якими циркулюють дані всередині проектованої інформаційної системи, а також між шляхами проходження інформації між системою і зовнішнім світом.

Будь-який DFD починається з оглядового DFD, у якому коротко описується проектована система – так званий верхній контекстний рівень (верхневрівнева контекстна діаграма).

Логічна діаграма потоку даних зосереджена на бізнесі та розповідає про події, що відбуваються у бізнесі, та даних, що генеруються в результаті кожної такої події.

Тоді як *фізична діаграма потоку даних* представляє те, як має бути представлений потік інформації в системі.

Спільно логічна та фізична діаграми потоків даних можуть повністю візуалізувати поточний стан інформаційної (та іншої) системи та моделювати новий стан, який необхідно розглянути, а потім реалізувати.

На практиці широко поширене використання DFD для представлення логічного потоку даних та обробки даних.

Будь-яка діаграма потоку даних (DFD) відображає потік інформації для процесу або системи, тоді як логічна діаграма надає "що" відбувається, а фізична – "як" це відбувається. Це дві різні точки зору на той самий потік даних, кожна з яких призначена для візуалізації та уточнення системи.

Логічна DFD – зображує потоки даних. Такі діаграми показують переміщення потоку даних, життєво важливих для функціонування організації. У центрі уваги таких діаграм — сам бізнес та необхідна йому інформація, а не те, як працює чи має працювати система. Фокус логічний DFD – бізнес та ділова активність. Перевага логічних DFD полягає в тому, що вони легко сприймаються та читаються «не фахівцями». Такі моделі – це добрий інструмент обміну інформацією.

Фізична DFD - Зображує потоки фізичних сутностей. На відміну від логічних DFD, фізичні діаграми DFD ілюструють саме поточний або запланований пристрій системи. Наприклад, у логічній діаграмі DFD як процесів виступають різні види комерційної діяльності, а фізичної — програми та ручні процедури. Фізична DFD дивиться на те, як реалізована система.

Методологія діаграм потоків даних (DFD) складається із чотирьох елементів:

– зовнішніх сутностей,
– процесів,
– сховищ даних
– потоків даних.

Однак, елементи представляють різні точки зору в логічних DFD і фізичних DFD, а також можуть мати різне візуальне відображення в нотаціях.

Процес – у логічних процесах DFD є бізнес-операції, а у фізичних процесах DFD є програми, ручні процедури або інші способи обробки інформації.

Сховище даних – у логічних DFD сховищах даних є набори інформації, незалежно від цього, як вони зберігаються, а фізичних DFD сховищами даних є бази даних, комп'ютерні файли і паперові файли.

Процес (process) / *робота* (activity) – функція або послідовність дій, які потрібно зробити, щоб дані були оброблені. Це можливо створення замовлення, реєстрація клієнта тощо. У назви процесів прийнято використовувати дієслова, тобто, "Обробити замовлення" (а не "Проведення замовлення"). Тут немає суворої системи вимог, як, наприклад, IDEF0 або BPMN, де нотації мають жорстко певний синтаксис, так як вони можуть бути виконуваними. Але все ж таки певних правил варто дотримуватися, щоб не вносити плутанину під час читання DFD іншими людьми.

Зовнішні сутності / *посилання* (external entity / external reference). Це будь-які об'єкти, які не входять до самої системи, але є для неї джерелом інформації або одержувачами будь-якої інформації із системи після обробки даних. Це може бути людина, зовнішня система, будь-які носії інформації та сховища даних.

Сховище даних (data store) – внутрішнє сховище даних для процесів у системі. Дані перед обробкою і результат після обробки, а також проміжні значення повинні десь зберігатися. Це і є бази даних, таблиці чи будь-який інший варіант організації та зберігання даних. Тут зберігатимуться дані про клієнтів, заявки клієнтів, видаткові накладні та будь-які інші дані, що надійшли до системи або є результатом обробки процесів.

Потік даних (data flow) – в нотації відображається у вигляді стрілок, які показують, яка інформація входить, а яка виходить з того чи іншого блоку на діаграмі.

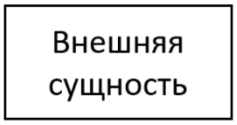
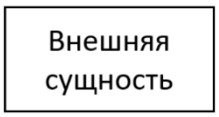
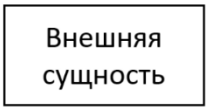
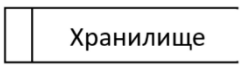
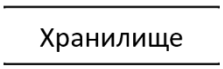
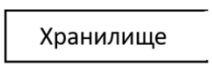
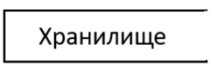
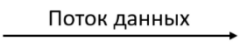
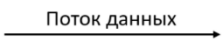
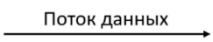
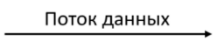
2. Варіанти графічних нотацій

Діаграми потоків даних стали відомі широкому загалу з кінця 1970-х років завдяки книзі «Структурне проектування» піонерів обчислювальної техніки Еда Йордана та Ларрі Костянтина («Structured Design» Yourdon & Constantine, 1974).

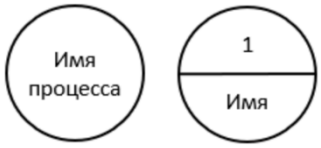
Найбільш поширені нотації:

– Peter Coad and Ed Yourdon – методологія Коада та Йордану.
– Ed Yourdon and Tom DeMarco (Yordon-DeMarco notation) - нотація Йордон-ДеМарко.

– Chris Gane та Trish Sarson (Gene-Sarson DFD Symbols or notation) – нотація Гейна-Сарсона.
– SSADM (Structured System Analysis and Design Methodology).

Символ / Назначение	Гейна-Сарсона (Gene-Sarson)	Йордона-ДеМарко (Yordon-DeMarco)	Коад и Йордон (Coad and Yordon)	SSADM
Функция / Процесс Работа, действие, операция преобразования данных. Выполняет какие-либо действия над данными, как например: создает, модифицирует, сохраняет, удаляет и т.д.				
Внешняя сущность Является источником или адресатом потока данных. Отображает внешние по отношению к системе сущности.				
Хранилище данных Используется для хранения данных				
Поток данных Объект. Потоки данных между процессами, хранилищами, внешними сущностями.				

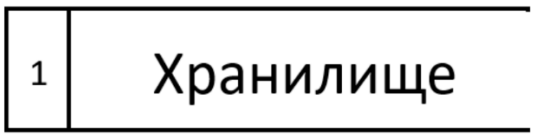
Розглянемо приклади відображення процесів:

Гейна-Сарсона (Gene-Sarson)	Йордона-ДеМарко (Yordon-DeMarco)
	

Зовнішні сутності – це будь-які об'єкти, які входять у саму систему, але є для неї джерелом інформації чи одержувачами будь-якої інформації із системи після обробки даних. Наприклад, матеріальний об'єкт або фізична особа, яка є джерелом або приймачем інформації (замовники, клієнти, постачальники, склад, персонал, банк), а також зовнішня система, будь-які носії інформації та сховища даних. Зовнішня сутність завжди знаходиться поза межами аналізованої системи. Одна і та ж зовнішня сутність може бути використана багаторазово на одній або кількох діаграмах.

Гейна-Сарсона (Gene-Sarson)	Йордона-ДеМарко (Yordon-DeMarco)
	

Сховище даних – це елемент, що представляє внутрішнє сховище для процесів у системі.

Гейна-Сарсона (Gene-Sarson)	Йордона-ДеМарко (Yordon-DeMarco)
	

Потік даних – це елемент, який визначає інформацію, яка передається через деякі з'єднання від джерела до приймача. У нотації відображається у вигляді стрілок, які показують, яка інформація входить, а яка виходить із того чи іншого блоку на діаграмі.



Контекстна діаграма – це важливий елемент методології DFD, а саме це коренева діаграма, на якій зображено:

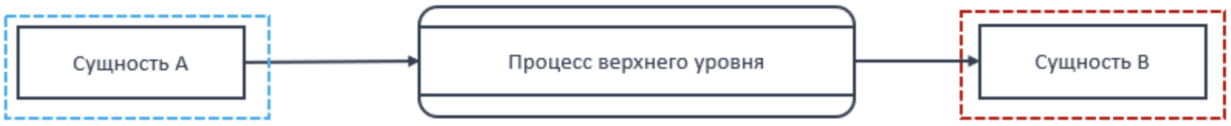
– Усі «зацікавлені» зовнішні сутності, які спілкуються із системою.

– Єдиний процес із номером «0», який зображує всю систему цілком.

– Основні потоки даних між системою та зовнішнім світом.

Зазвичай DFD малюють ітераційно, тому що складно відразу визначити всі потоки даних та ідентифікувати всі зовнішні сутності за один раз.

Наприклад:

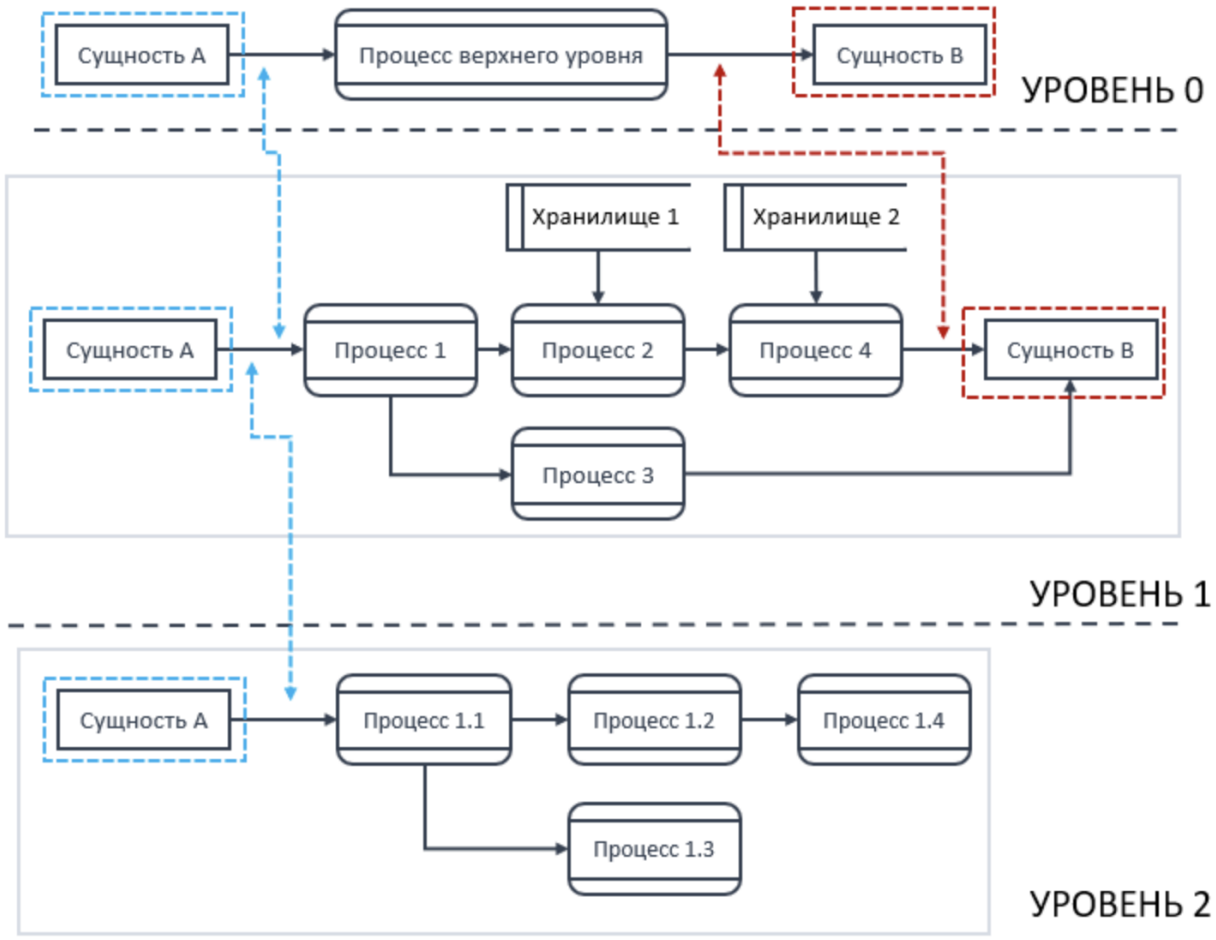


Рівні DFD-моделі (або декомпозиції) можна наочно у вигляді наступної схеми:



Більш детально вони зображаються так:

– Перша діаграма називається контекстною (рівень 0) і служить позначення рамок системи (завдання контексту, у якому система працює).
– Діаграма деталізується шляхом додавання нових рівнів.
– На кожному рівні декомпозується (будується окрема діаграма) один якийсь процес із попереднього рівня.



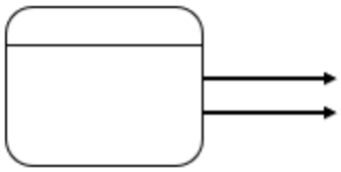
Нумерація елементів на діаграмі – є одне важливе поняття. Як і в IDEF0 нумерація функціональних блоків має ієрархічний характер:

Рівень 0 – 0
Рівень 1 – 1, 2, 3, …
Рівень 2 – 1.1, 1.2, …, 2.1, 2.2, …
Рівень 3 – 1.1.1, 1.1.2, …, 1.2.1, 1.2.2, …
тощо.

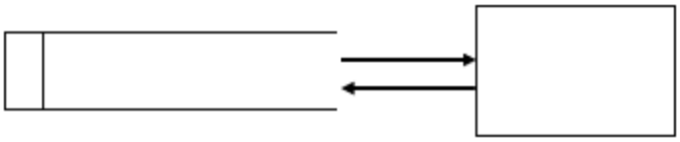


2. Приклади використання DFD

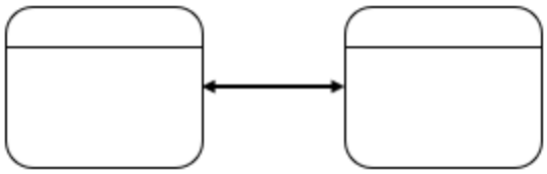
Поширені помилки під час використання нотацій методології DFD:
У процесу є потоки, що виходять, але немає вхідних.



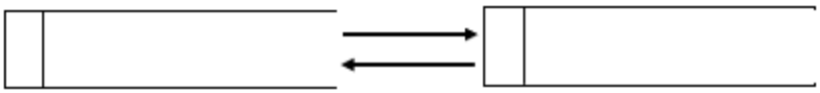
Сховище та зовнішнє джерело пов'язані безпосередньо.



Потік йде прямо у двох напрямках.

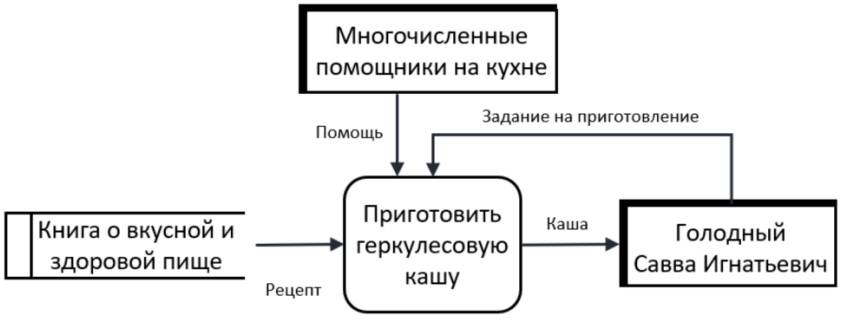


Сховища пов'язані безпосередньо.

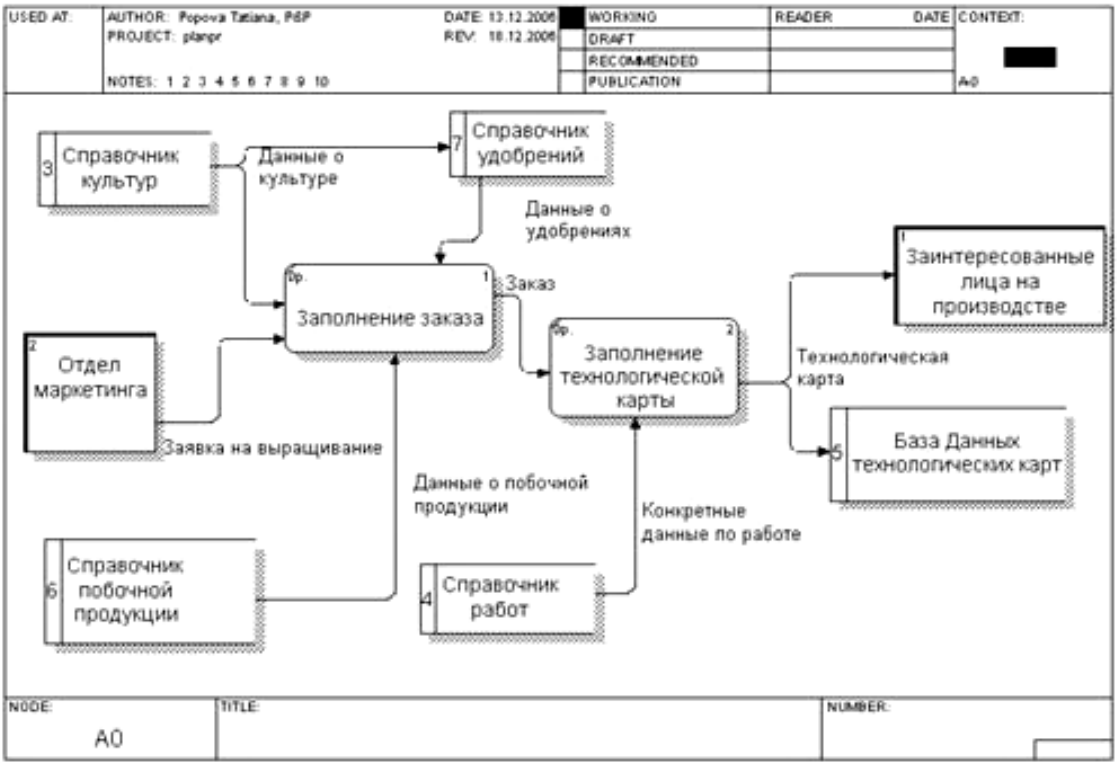


Приклад 1.

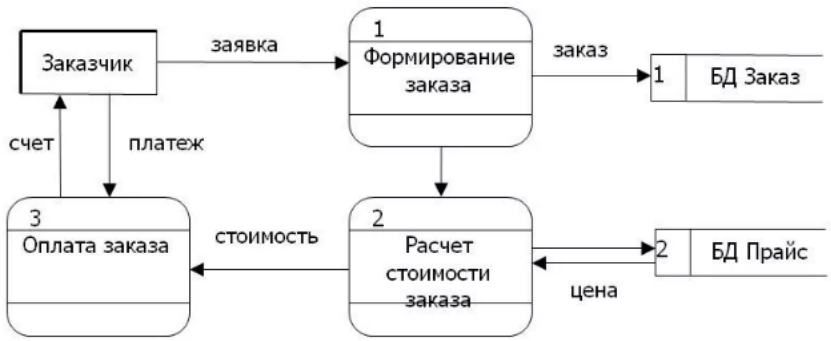
Нотация Гейна-Сарсона (Gene-Sarson)



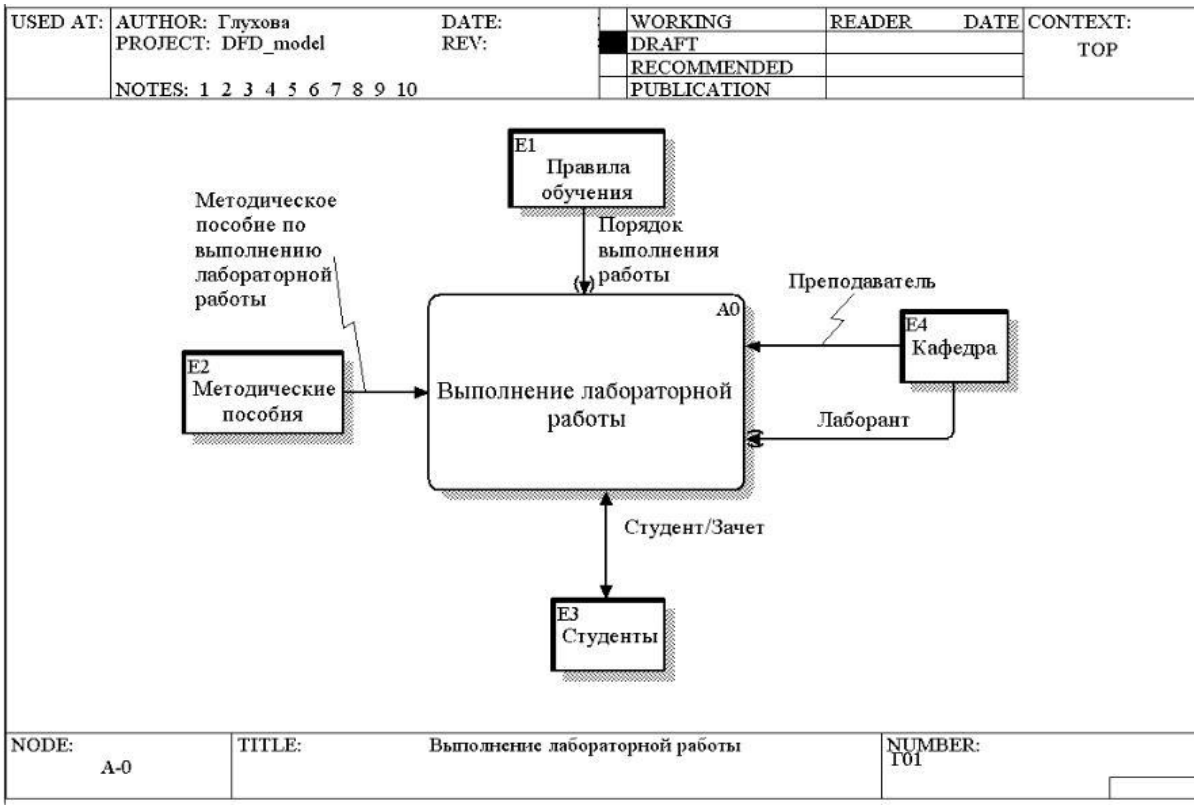
Пример 2.



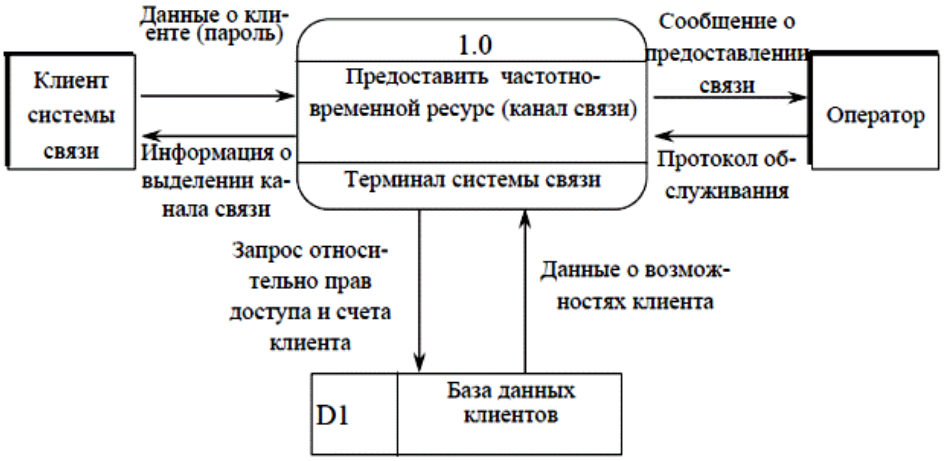
Пример 3.



Приклад 4.



Приклад 5.



Запитання для самоперевірки

1. Для опису чого використовується нотація DFD?
2. З яких чотирьох елементів складається методологія діаграм потоків даних?
3. Чим відрізняються логічна DFD від фізичної DFD?
4. Які варіанти графічних нотацій ви знаєте? Чим вони відрізняються.
5. Які поширені помилки виникають під час використання нотацій DFD?

Лекція 10. Нотація EPC

1. Компоненти синтаксису EPC
2. Приклади використання DFD

1. Компоненти синтаксису EPC

Подієвий ланцюжок процесів (EPC-діаграма, англ. event-driven process chain) – тип блок-схеми, що використовується для бізнесу-моделювання. EPC може бути використана для налаштування системи планування ресурсів підприємства. Організації використовують EPC діаграми для планування потоків робіт бізнес-процесів.

EPC-метод було розроблено Августом-Вільгельмом Шеером в рамках робіт над створенням ARIS на початку 1990-х років. Основна мета – надати таку систему опису процесів, де всі функції будуть семантично пов'язані. Система не передбачає обов'язкового чіткого прописування виконання функцій, основну увагу звертають на події.

Нотація EPC (Event-Driven Process Chain – подієвий ланцюжок процесів) використовується для опису процесів нижнього рівня. Діаграма процесу в нотації EPC є упорядкованою комбінацією подій і функцій.

Для кожної функції можуть бути визначені початкові та кінцеві події, учасники, виконавці, матеріальні та документальні потоки, які її супроводжують, а також проведена декомпозиція на нижчі рівні. Декомпозиція може виконуватись у нотаціях EPC або BPMN.

EPC-діаграми використовують символи декількох видів, щоб показати структуру потоку управління (послідовність рішень, функції, події та інші елементи) бізнес-процесу.

У нотації EPC використовується значна кількість елементів, які представлені різнокольоровими фігурами. Рожевими фігурами відображають *події*, зеленими – *дії*, жовтими – *виконавців*, сірими – *ресурси*, помаранчевими – *інформаційні системи*.

Прийнято розгортати модель зверху вниз, коли ранні елементи відображаються зверху.

Крім стрілок – з'єднувальних елементів у нотації передбачено використання роздільників «і», «або» та «що виключає або». Це дає можливість використовувати цю методику при описі бізнес-процесів, що розгалужуються.

Схема містить упорядковану комбінацію подій та функцій. Кожна функція повинна мати початкову та кінцеву подію, також можливе відображення учасників, виконавців, необхідних для виконання функції ресурсів.

Нотація допускає проведення декомпозиції на більш низьких рівнях.

Події є пасивними елементами в EPC.

Подія є стан, який зустрічається перед або після функції, тобто фіксує стан певних параметрів на певний момент часу. Приклади подій: "договір підписаний", "вимога зафіксована", "матеріал на складі".

Функції є активними елементами в EPC.

Робота (функції) – певна дія, що виконується протягом деякого проміжку часу. Кожна робота може бути декомпозована.

Організаційна одиниця – посада в організації (наприклад, «старший майстер») або підрозділ організації (наприклад, «відділ закупівель»), елемент, якому може бути доручено виконання функції.

Інформація, *матеріал*, або *об'єкт ресурсу* – об'єкти в реальному світі, наприклад бізнес-об'єкти, різні сутності, які можуть бути як вхідними даними, що виступають як основа функції, так і вихідними даними, отриманими за допомогою функції. Прикладами є "матеріал", "замовлення".

Логічний з'єднувач – елемент управління в діаграмі, що визначає розгалуження потоку робіт в залежності від завершення виконання функції або виникнення подій.

Логічні взаємозв'язки – елементи управління, що відповідають за зчленування потоків - кон'юнкція, диз'юнкція або строга диз'юнкція.

Потік управління поєднує події з функціями, шляхами процесу або логічними взаємозв'язками, створюючи хронологічну послідовність або логічну взаємозалежність між ними.

Потік інформації – з'єднання функції та вхідних та вихідних даних, з яких функція зчитує зміни або сама їх вносить.

Призначення організаційної одиниці – зв'язок між організаційною одиницею та функцією, за яку вона відповідальна.

Шлях процесу – елемент, що показує взаємозв'язок з іншими процесами.

Розглянемо алгоритм побудови діаграми:

– Визначаємо початкову та кінцеву події, іншими словами, проводимо оцінку поточного та бажаного стану.
– Впроваджуємо між граничними подіями дії (або функції) та відповідні проміжні події.
– Визначаємо та відображаємо на схемі документи та інформацію, необхідну для реалізації кожного з етапів, а також результуючі документи. Також визначаємо виконавців робіт, позначаємо їхню роль.
– Аналізуємо якість та повноту схеми, розглядаємо, чи всі можливі варіанти подій та дій передбачені моделлю. Якщо виявлено функції, які зовсім зрозумілі, вимагають розшифровки, деталізуємо в підпроцеси і розробляємо відповідні схеми їм.

Функція



дія або набір дій, що виконуються над вихідним об'єктом (документом, ТМЦ та іншим) з метою отримання заданого результату. Усередині блоку міститься найменування функції.

Тимчасова послідовність виконання функцій визначається розташуванням функцій на діаграмі процесу зверху вниз.



Подія



стан, що є суттєвим для цілей управління бізнесом та впливає або контролює подальший розвиток одного або більше бізнес-процесів. Елемент відображає події, що активізують функції, що породжуються функціями. Усередині блоку міститься найменування події.



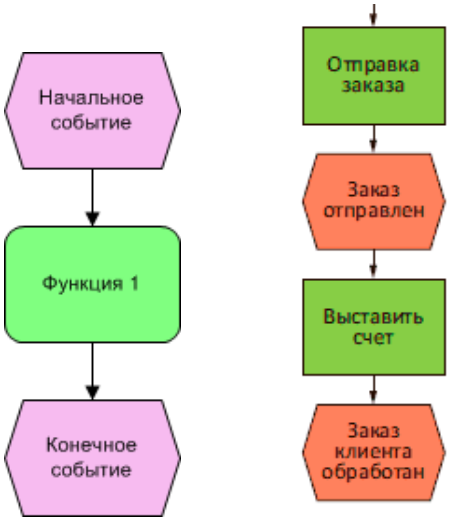
Стрілка



відображає зв'язки елементів діаграми EPC процесу між собою.

Зв'язок може бути спрямованим і не спрямованим залежно від елементів, що з'єднуються, і типу зв'язку.

Наприклад:



Суб'єкт



використовується відображення на діаграмі організаційних одиниць (посади, підрозділи, ролі, зовнішнього суб'єкта) - виконавців, власників чи учасників функцій. Усередині блоку міститься найменування організаційної одиниці.

Наприклад:



Паперовий документ



використовується для відображення на діаграмі паперових документів, що супроводжують виконання функції. Усередині блоку міститься найменування паперового документа.

Наприклад:

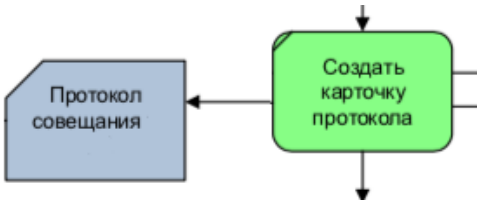


Електронний документ

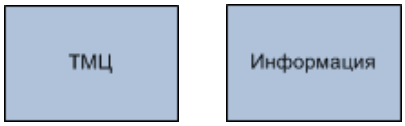


використовується для відображення діаграми електронних документів, що супроводжують виконання функції. Усередині блоку міститься найменування електронного документа.

Наприклад:

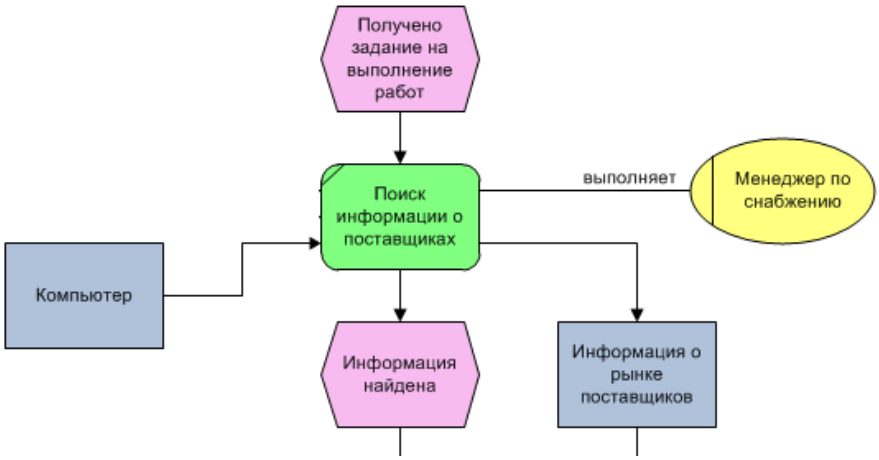


ТМЦ / Інформація



використовується для відображення на діаграмі товарно-матеріальних цінностей (ТМЦ) інформаційних потоків, що супроводжують виконання функції. Усередині блоку міститься найменування інформаційного потоку або ТМЦ.

Наприклад:



Інформаційна система / Модуль інформаційної системи / Функція інформаційної системи позначається наступним чином



Наприклад:

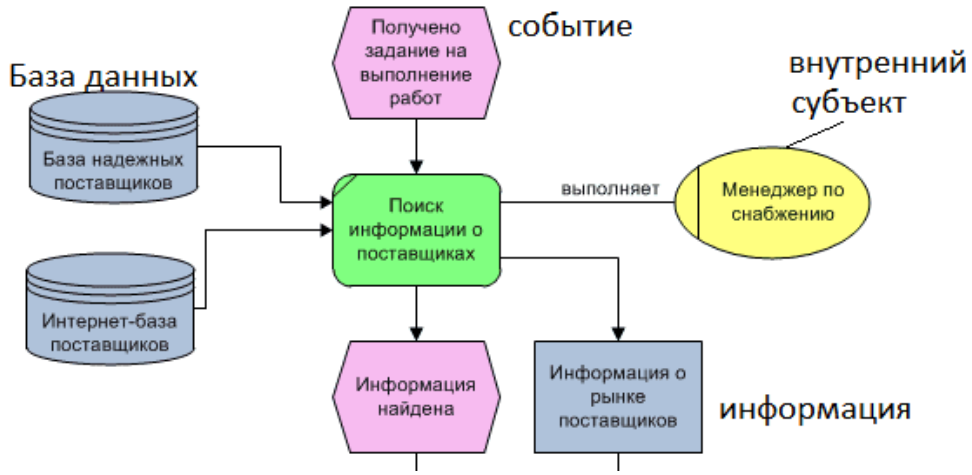


База даних

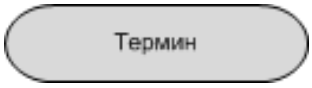


використовується для відображення діаграми бази даних, що супроводжує виконання функції. Усередині блоку міститься найменування бази даних.

Наприклад:



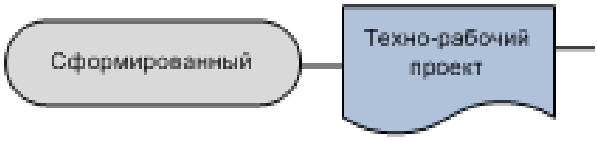
Термін / Статус об'єкта



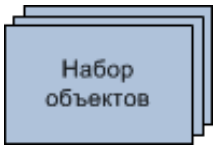
елемент може бути використаний для позначення даних, що передаються між процесами або оброблюваних при виконанні процесів. Елемент може бути також використаний для позначення статусів паперових/електронних документів та інших елементів

Наприклад:

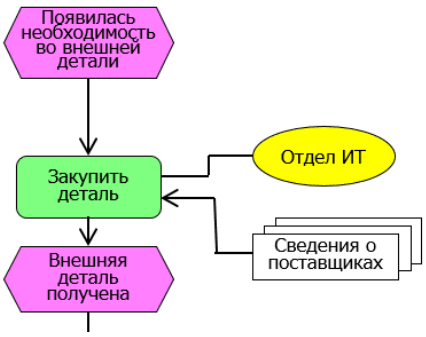




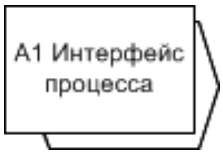
Набір об'єктів



Наприклад:



Інтерфейс процесу

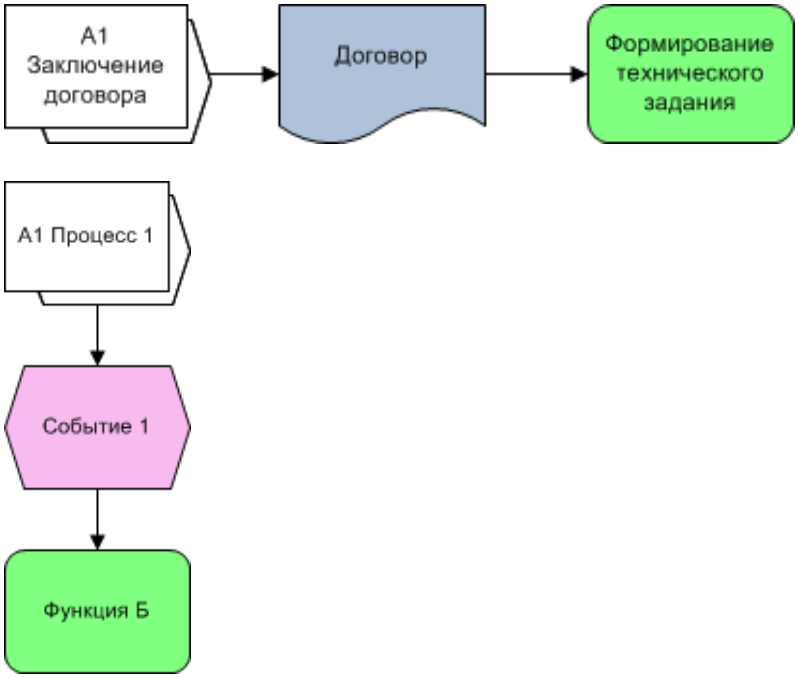


елемент, що позначає зовнішній (стосовно поточної діаграми) процес або функцію. Усередині блоку міститься найменування зовнішнього процесу.

Використовується для зазначення взаємозв'язку процесів:

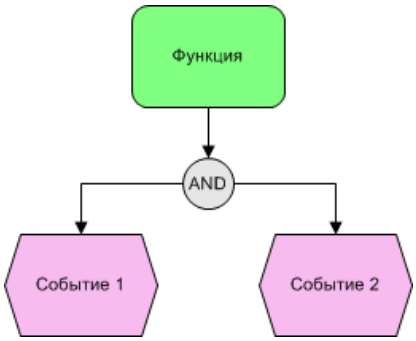
– позначає попередній або наступний процес по відношенню до діаграми процесу;
– позначає процес, звідки надійшов або куди передається об'єкт.

Наприклад:

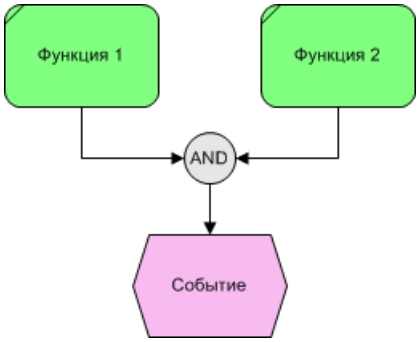


Оператор "І" 

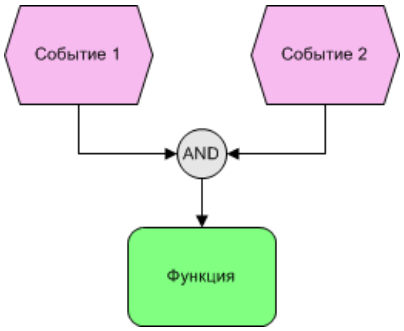
використовується для позначення злиття/розгалуження як функцій, так і подій. Якщо завершення виконання функції має ініціювати одночасно кілька подій, це позначається за допомогою оператора "І", наступного після функції і перед подіями.



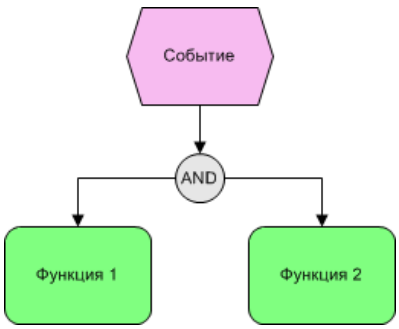
Якщо подія відбувається лише після обов'язкового завершення виконання кількох функцій, це позначається за допомогою оператора "І", наступного після функцій і перед поодинокою подією.



Якщо функція може почати виконуватися тільки після того, як відбудуться кілька подій, це позначається за допомогою оператора "І", наступного після подій і перед функцією.



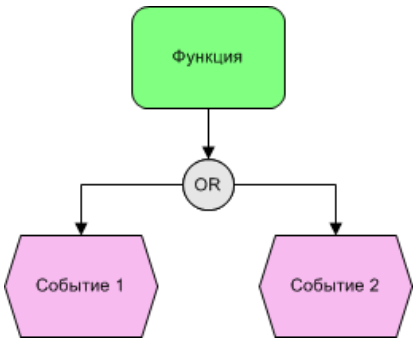
Якщо одна подія може ініціювати одночасне виконання кількох функцій, це позначається за допомогою оператора "І", наступного після події і перед функціями.



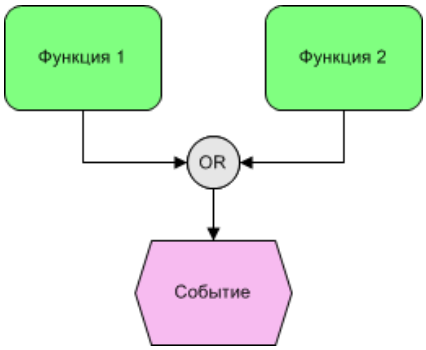
Оператор "АБО" 

використовується для позначення злиття/розгалуження функцій та для злиття подій. За правилами нотації EPC після одиночної події не може наслідувати оператор "АБО", що розгалужує.

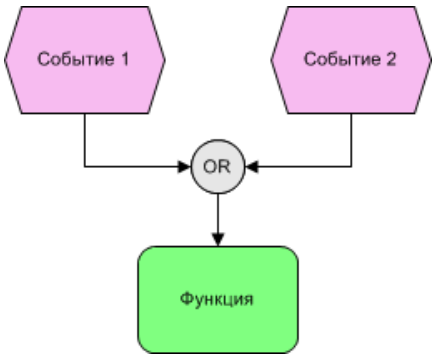
Якщо завершення виконання функції може ініціювати одну або кілька подій, це позначається за допомогою оператора "АБО", наступного після функції та перед подіями.



Якщо подія відбувається після завершення виконання однієї або кількох функцій, це позначається за допомогою оператора "АБО", наступного після функцій і перед поодинокою подією.



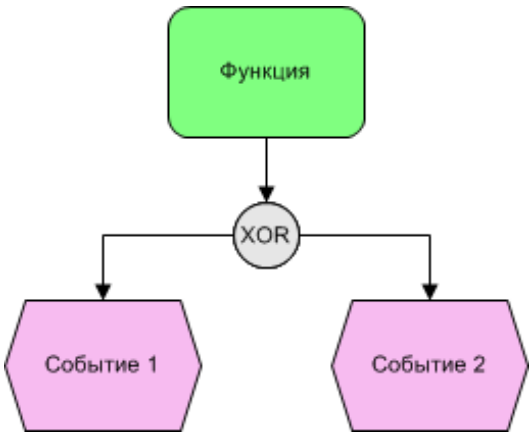
Якщо функція може почати виконуватися після того, як відбудеться одна або кілька подій, це позначається за допомогою оператора "АБО", наступного після подій і перед функцією.



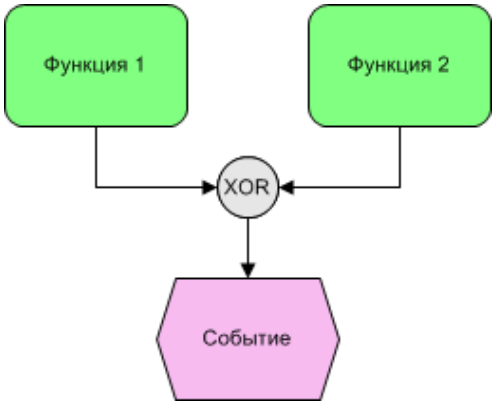
Оператор "Виключне АБО" 

використовується для позначення злиття/розгалуження функцій та для злиття подій. За правилами нотації EPC після одиночної події не може слідувати розгалужувальний оператор "Виключне АБО".

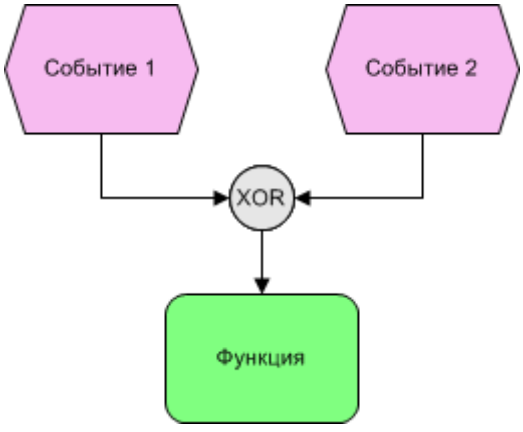
Якщо завершення виконання функції може ініціювати лише одну з подій залежно від умови, це позначається за допомогою оператора "Виключне АБО", наступного за функцією і перед подіями.



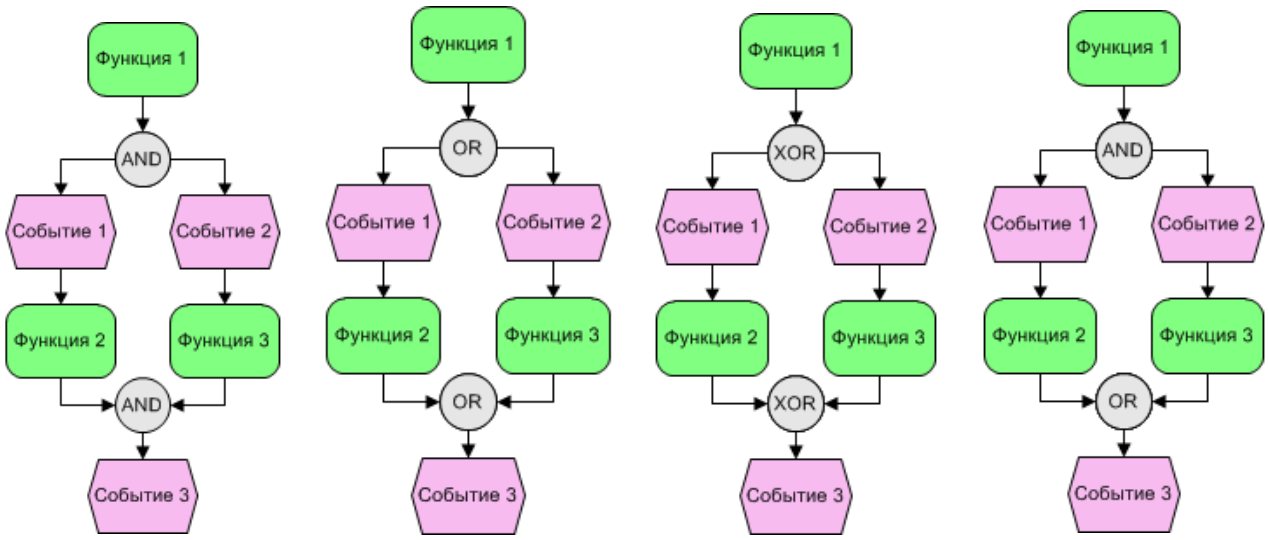
Якщо подія відбувається відразу після завершення виконання однієї функції, або іншої, то це позначається за допомогою оператора "Виключне АБО", наступного після функцій і перед одиночною подією.



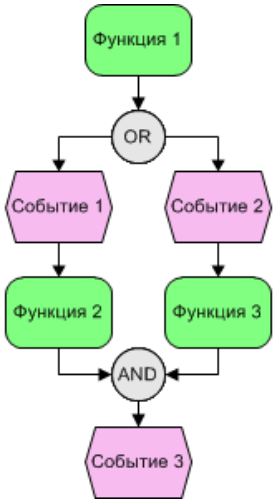
Якщо функція може почати виконуватися відразу після того, як відбудеться або одна подія, або інша, то це позначається за допомогою оператора "Виключне АБО", наступного після кількох подій та перед функцією.



Приклади допустимих ситуацій:



Приклад неприпустимої ситуації:

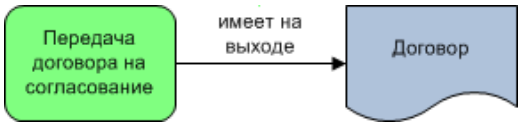


Типи зв'язків, які можна навести між елементами на діаграмі EPC:

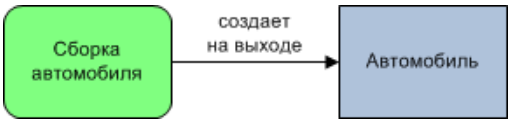
– змінює



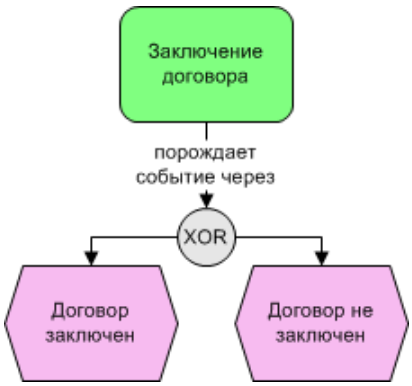
– має на виході



– створює на виході



– має, породжує подію через xor



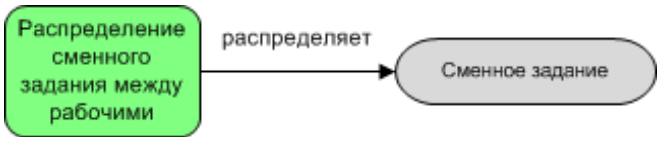
– передує



– породжує



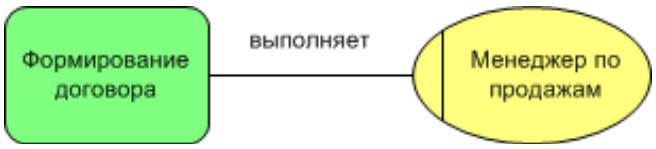
– розподіляє



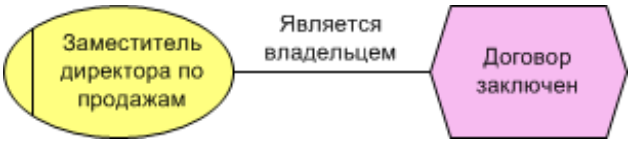
– зчитує



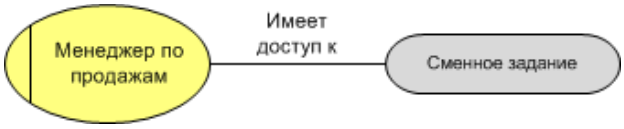
– виконує



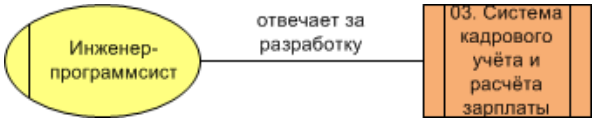
– є власником



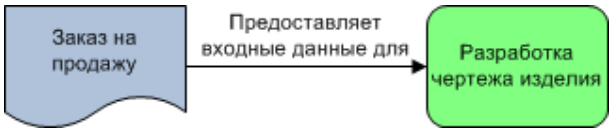
– має доступ до



– відповідає за розробку



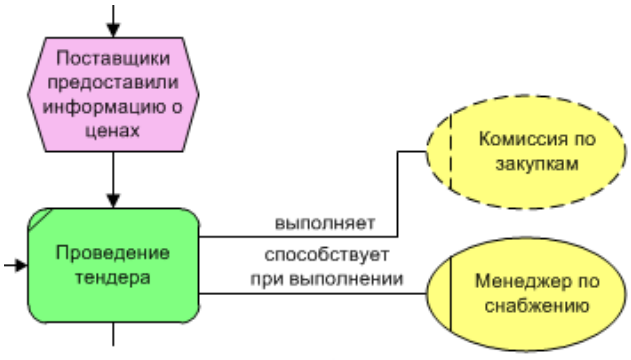
– надає вхідні дані для



– встановлює статус



Наприклад:



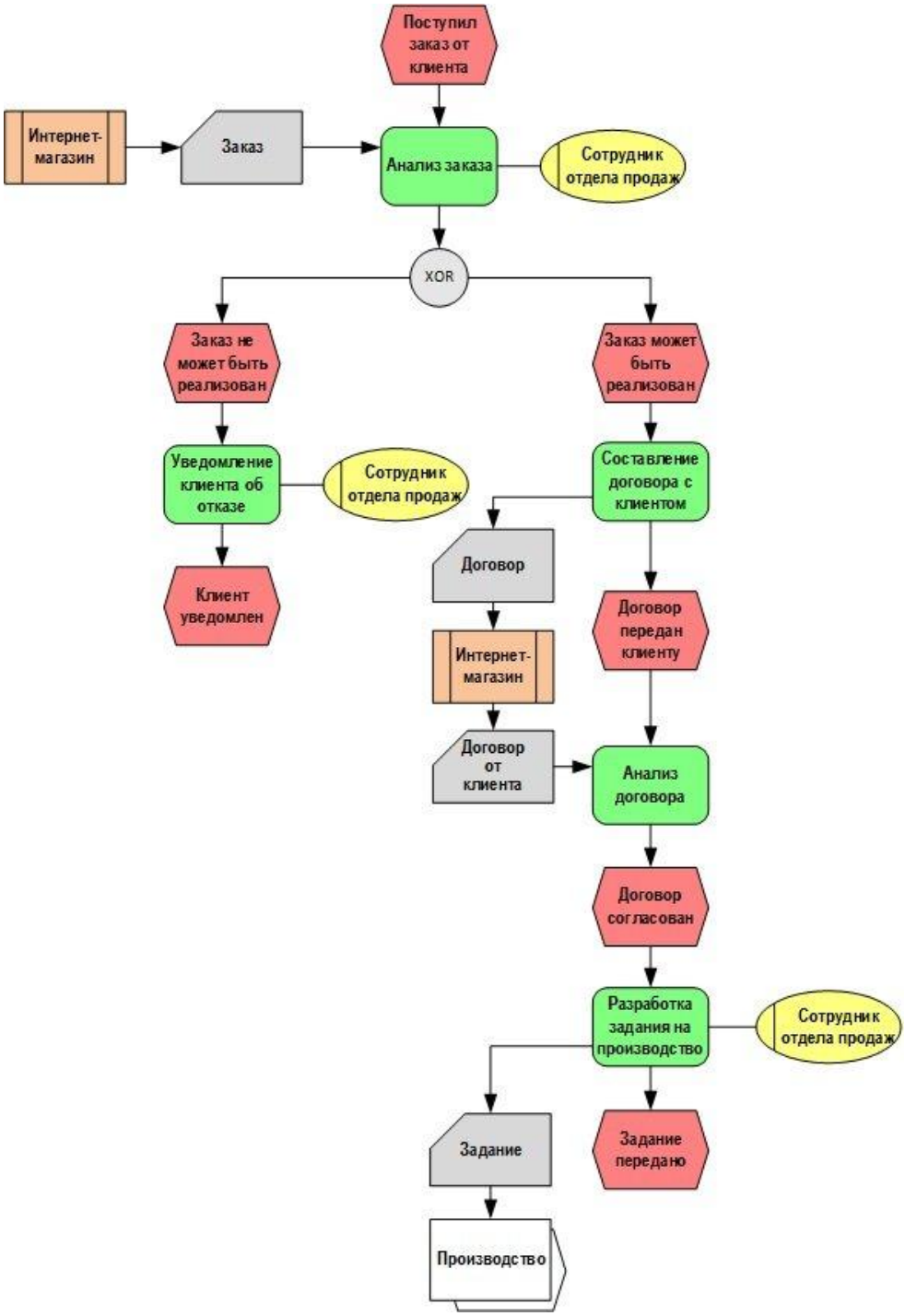
Приклад загальні правила, дотримання яких дозволить досягти єдності у створенні та оформленні моделі різними фахівцями:

– Переважно розміщення схеми зверху донизу. Допустимо - зліва направо.
– Елементи виконавців маємо праворуч від функцій.

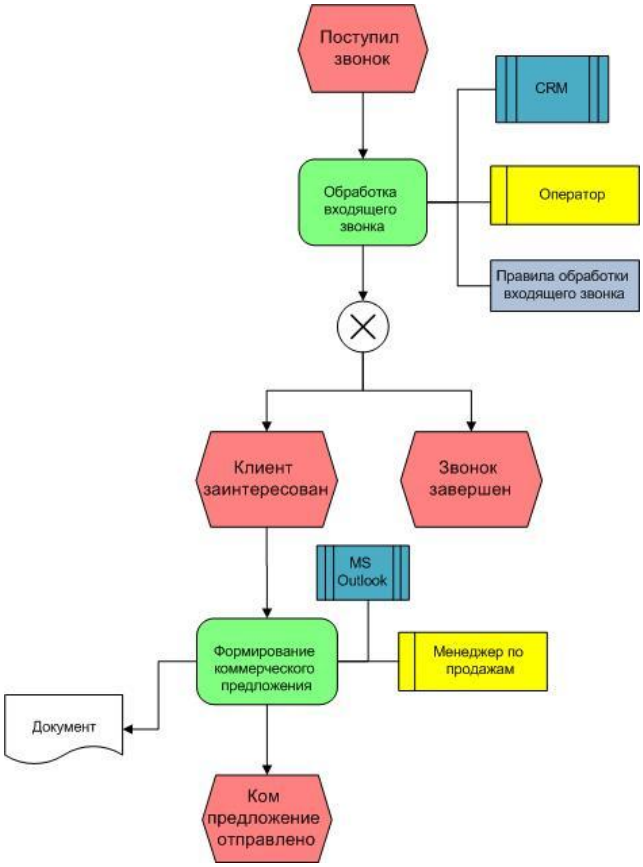
– Ліворуч та вгору від функції даємо інформацію про вхідні документи.
– Вліво та вниз від функції представляємо інформацію про вихідні документи.
– Вниз і вправо від функції маємо блок «Інформація».
– Вгору та вправо від функції представляємо дані про програми.
– Довільне розташування переважно для елементів «База даних» та «Картотека».
– Дані про матеріальні потоки подаємо зліва про відомості про документи, що їх супроводжують.

2. Приклади використання DFD

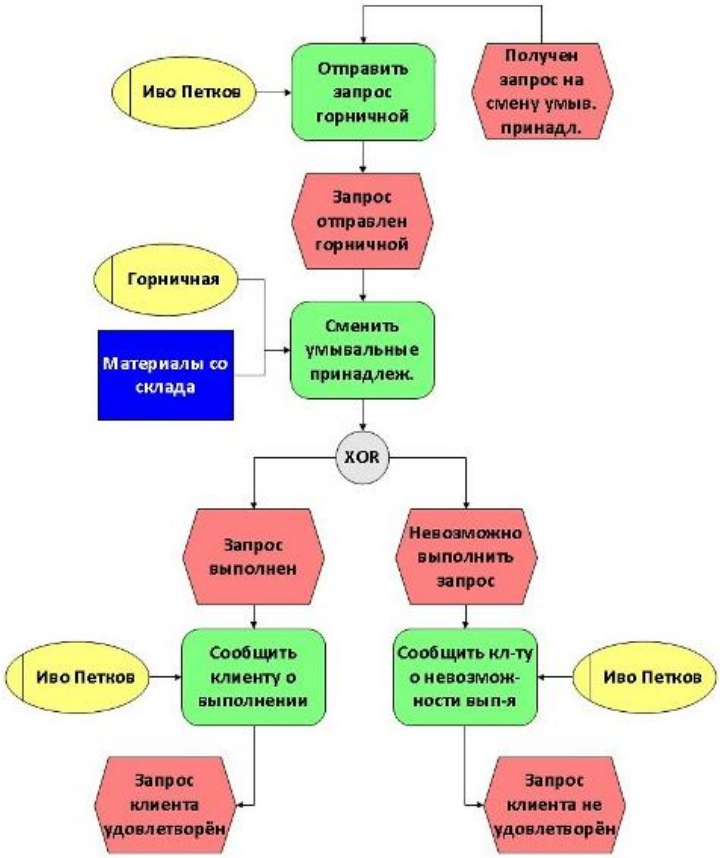
Розглянемо приклад:



Приклад:



Приклад процесу обробки запиту клієнта в готелі:



Запитання для самоперевірки

1. Для опису чого використовується нотація EPC?
2. З яких елементів складається нотація EPC?
3. Для чого використовуються оператори «І», «АБО», «Виключне АБО»?
4. Які типи зв'язків, існують між елементами на діаграмі EPC?

Література

1. Federal Information Processing Standards Publication 183. Integration definition for function modeling (IDEF0). Dec. 1993
2. Federal Information Processing Standards Publication 184. Integration definition for information modeling (IDEF1X). Dec. 1993
3. Моделювання бізнес-процесів. URL: https://stud.com.ua/87161/ekonomika/ modelyuvannya_biznes-protsesiv
4. Лекція 04. Методологія структурного аналізу та проектування SADT. URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?id=39227
5. Інструменти управління та моделювання бізнес процесів. URL: https://rzbpm.ru/knowledge/instrumenty-upravleniya-i-modelirovaniya-biznes-processov.html
6. Посібник для початківців із використання нотації BPMN у повсякденній роботі. URL: https://www.microsoft.com/uk-ua/microsoft-365/business-insights-ideas/resources/the-guide-to-using-bpmn-in-your-business
7. Нетепчук В.В., Управління бізнес-процесами: Навч. посібник. – Рівне: НУВГП, 2014. – 158 с
8. Рєпін В.В. Процесний підхід до управління / Рєпін В.В., Елиферов В.Г.,. – М.: РИА «Стандарты и качество», 2007. – 408 с.
9. Грекул В.И. Проектування інформаційних систем / Грекул В.И., Денищенко Г.Н., Коровкина Н.Л. — М.: Інтернет-університет інформаційних технологій, 2005. — 304 с.
10. Буч Г. Язык UML. Керівництво користувач/ Г.Буч, Д.Рамбо, И.Якобсон. – М.: ДМК Пресс, 2006. – 496 с.
11. Araújo M.B. Selecting a Notation to Modeling Business Process: A Systematic Literature Review of Technics and Tools. / M.B.Araújo, R.F. Gonçalves // IFIP Advances in Information and Communication Technology. – 2016. – vol. 488. P. 198-205
12. Нотация IDEF0. URL: http://www.businessstudio.com.ua/bp/bs/overview/notation_idef0.php
13. Business Process Modeling Notation (BPMN). Version 1.2. – Object Management Group, 2009. – 316 p.
14. Brackett, J. Applying SADT to Large System Problems / J.Brackett, C. McGowan. – SofTech Technical Paper TP059,January 1977