Міністерство освіти і науки України
Державний університет «Одеська політехніка»

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт
з дисципліни «Бізнес-процеси в кібербезпеці»
другого (магістерський) рівня вищої освіти
спеціальності 125 «Кібербезпека»

Одеса, 2021

Міністерство освіти і науки України
Державний університет «Одеська політехніка»

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт
з дисципліни «Бізнес-процеси в кібербезпеці»
другого (магістерський) рівня вищої освіти
спеціальності 125 «Кібербезпека»

Затверджено
на засіданні кафедри КБПЗ
Протокол № 1 від 27.08.2021 р.

Одеса, 2021

Методичні вказівки до лабораторних робіт з дисципліни «Бізнес-процеси в кібербезпеці» для другого (магістерського) рівня вищої освіти спеціальності 125 – Кібербезпека / Укл.: *Лебедєва О.Ю.* – Одеса, 2021. – 55 с.

Укладач: Лебедєва О.Ю., к.т.н., доц.

ЗМІСТ

Семестровий модуль 1
Змістовий модуль 1. Методології моделювання бізнес-процесів
Лабораторні роботи № 1. Дослідження способів опису бізнес-процесів

Мета роботи:

Розглянути визначення бізнес-процесу та способів його опису. Навчитися описувати бізнес-процеси за допомогою розглянутих способів

Результати навчання:

Досліджувати, розробляти і супроводжувати системи та засоби інформаційної безпеки та/або кібербезпеки на об’єктах інформаційної діяльності та критичної інфраструктури.

1. Теоретичний матеріал

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.

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

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.

Способи опису бізнес-процесів:

– Текстовий
– Табличний
– Графічний

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

Цей спосіб підходить компаніям, які хочуть оптимізувати процеси «як є». З метою оптимізувати процеси «як має бути», текстовий спосіб опису не підходить. Суцільний текст не дозволяє подивитися на бізнес-процеси компанії системно та провести їх аналіз. Ще одним недоліком текстового методу є практично неможливість (або великі трудовитрати) внесення змін до документів, що регламентують.

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.

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).

В даний час найбільший розвиток і застосування при описі бізнес-процесів отримують графічні підходи і методи. Визнано, що вони мають найбільшу результативністю при

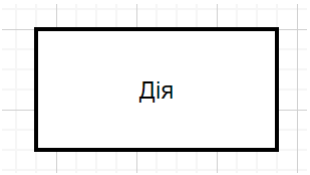
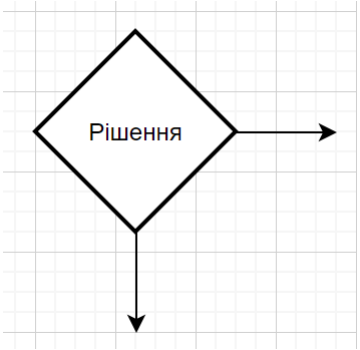
вирішенні задач по опису, аналізу та раціоналізації діяльності підприємства.

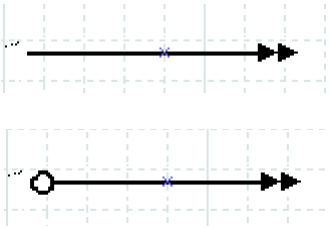
Блок-схема включає процес, представлений у вигляді сутностей (прямокутників довільної форми), пов'язаних відносинами (стрілками), які задають послідовність виконання функцій процесу.

Для опису блок-схеми можна використовувати нотацію Flowchart. Нотації Flowchart використовуються для представлення алгоритму (сценарію) виконання процесу і дозволяють задати причинно-наслідкові зв'язки та тимчасову послідовність виконання дій процесу.

Розглянемо призначення графічних символів нотації Flowchart (таблиця 1.1).

Таблиця 1.1 Графічні символи нотації Basic Flowchart

Графічний символ	Назва	Опис
	Подія	Події відображають стартові точки процесу в нотаціях Flowchart, що призводять до початку виконання процесу, і кінцеві точки, настанням яких закінчується виконання процесу. Початком процесу вважається подія, з якої виходять стрілки передачі управління. Кінець процесу вважається подія, в яку тільки входять стрілки передачі управління.
	Дія	Дія позначається прямокутним блоком. Усередині блоку міститься назва дії. Тимчасова послідовність виконання дій визначається розташуванням дій на діаграмі зверху вниз
	Рішення	Символ "Рішення" означає розгалуження, після якого процес може піти по одному і лише одному альтернативному напрямку залежно від певної умови. Символ "Рішення" може мати один або кілька входів та ряд альтернативних виходів. Якщо символ "Рішення" використовується для перевірки умови, то "Рішення" поміщається на діаграму після символу "Дія", у назві символу "Рішення" вказується умова, що перевіряється, а всі можливі варіанти значення умови показуються стрілками, що виходять. Якщо "Рішення" використовується для позначення дії, то всі можливі

		варіанти результатів цієї дії показуються стрілками, що виходять.
	Зв'язок передування	Стрілки "Зв'язок передування" позначають передачу управління від однієї дії в інший, тобто. попередня дія має закінчитись перш, ніж почнеться наступне. Стрілка, що запускає виконання дії, зображується зверху, що входить в дію. Стрілка, що позначає передачу управління іншому (іншому) дії, зображується знизу, що виходить з дії
	Потік об'єктів	Стрілки "Потік об'єктів" використовуються у випадках, коли необхідно показати, що з однієї дії об'єкти передаються до іншої, при цьому перша дія не запускає виконання другої. Стрілки "Потік об'єктів" позначаються стрілкою із двома трикутниками на кінці. Якщо позначення джерела об'єкта(ів) неважливе, такий об'єкт показується стрілкою з тунельованим початком Якщо джерелом об'єкта(ів) є одна з процесів процесу, то такий об'єкт показується за допомогою стрілки, що виходить з дії-джерела і входить в дію-споживач, для виконання якого необхідний об'єкт

Розглянемо опис бізнес-процесу «Здобувач здає лабораторну роботу викладачу», за допомогою способів опису, розглянутих вище.

Текстовий опис: «Викладач приймає лабораторну роботу у здобувача, ставить запитання, здобувач відповідає на запитання, викладач оцінює результати лабораторної роботи та відповіді на запитання з п’яти бальної шкали, якщо оцінка більше 2, то лабораторна робота є зданою».

Табличний спосіб:

№	Процес	Відпові-дальний	Вхід	Від кого	Вихід	Кому
1	Поява рішення здати лабораторну роботу (ЛР)	Здобувач	-	-		
2	Показ ЛР	Здобувач	Завдання	Викладач	Результати ЛР	Викладач

2	Прийом ЛР	Викладач	Результати ЛР	Здобувач	Оцінка за правильність виконання ЛР	Здобувач
3	Задання питань	Викладач	Результати ЛР	Здобувач	Питання	Здобувач
4	Відповідь на запитання	Здобувач	Питання	Викладач	Відповідь	Викладач
5	Оцінка ЛР	Викладач	Оцінка за правильність виконання ЛР, Відповідь	Здобувач	Оцінка	Здобувач
6	Порівняння оцінки з прохідною оцінкою	Викладач	Оцінка	Викладач	ЛР здана або ні	Здобувач

Графічний спосіб.

Для побудови діаграми нотації Basic Flowchart скористаємось онлайн додатком Draw.io (рисунок 1.1)

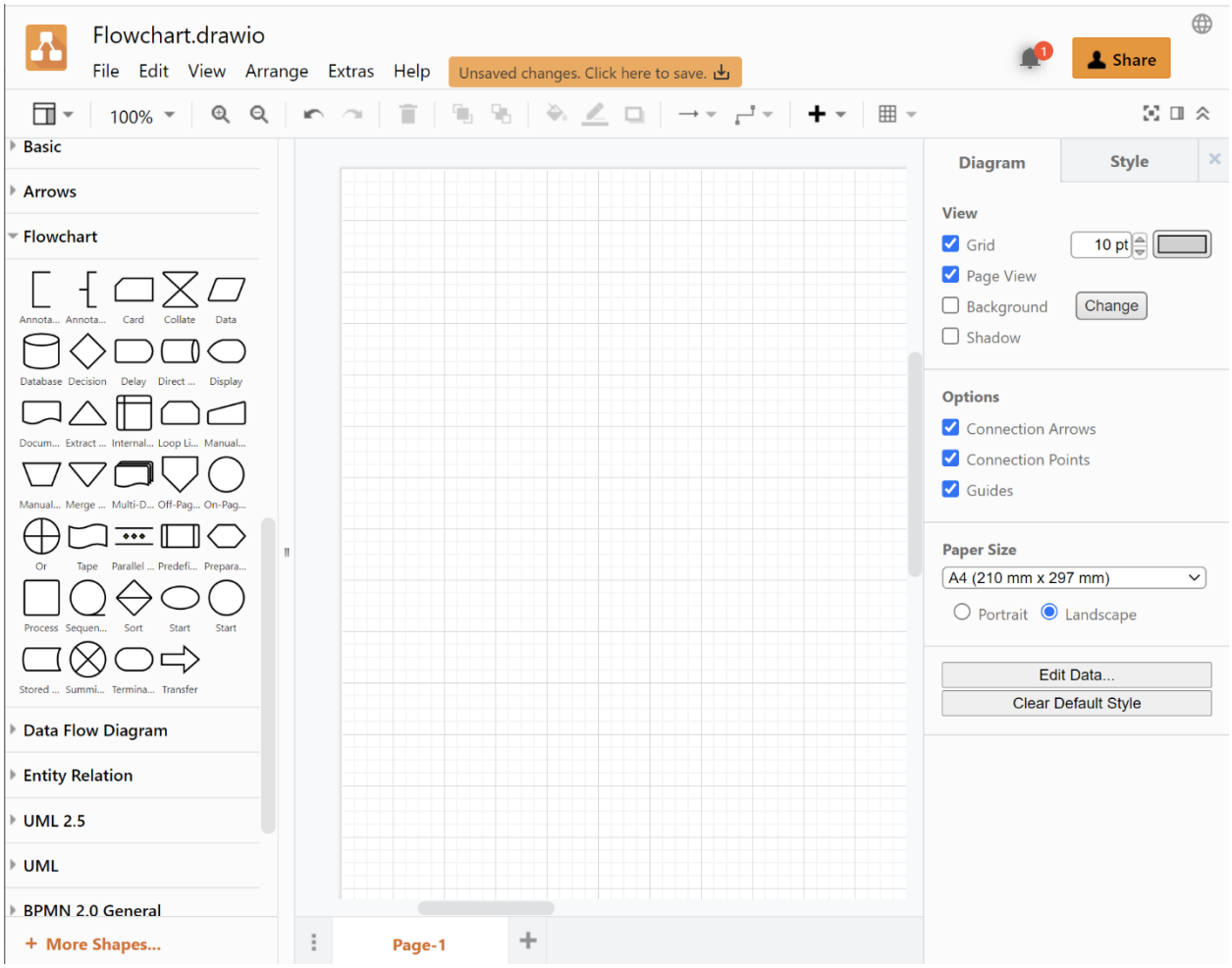


Рисунок 1.1 – Онлайн додаток Draw.io

Додаток Draw.io має в своєму арсеналі графічні символи нотації Flowchart. Якщо при відкритті додатку в списку груп графічних символів не має «Flowchart», то її можна включити натиснувши на кнопку «More shapes».

Результат запису поставленої задачі в нотації Flowchart продемонстровано на рисунку 1.2.

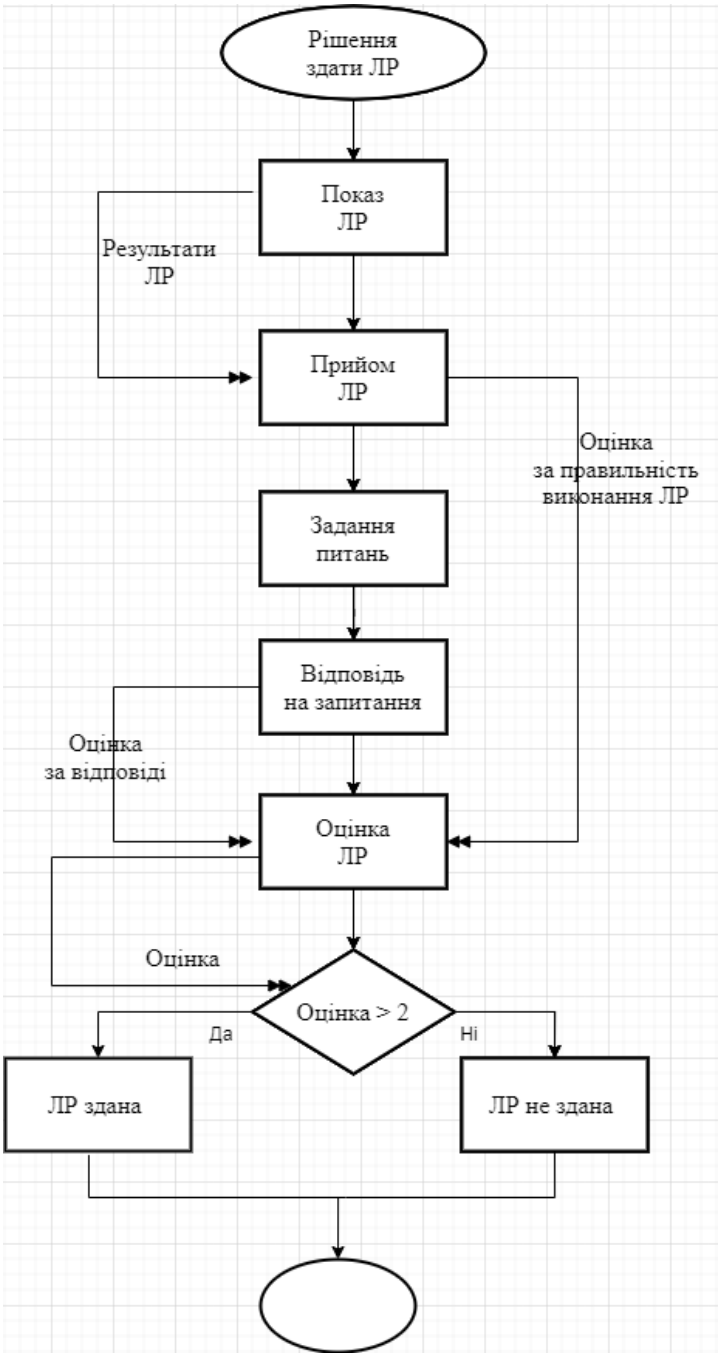


Рисунок 1.2 – Результат задачі в нотації Flowchart

2. Завдання до лабораторних робіт № 1

Оберіть задачу з області кібербезпеки, захисту інформації або іншої діяльності. Сформулювати обрану задачу. Описати її використовуючи текстовий, табличний та графічний спосіб.

Лабораторні роботи № 2. Дослідження структури та умов використання методології SADT

Мета роботи:

Розглянути визначення бізнес-процесу та спосіб його опису за допомогою методології SADT. Навчитися розробляти задачі у вигляді бізнес-процесу, описаного за методологією SADT.

Результати навчання:

Досліджувати, розробляти і супроводжувати системи та засоби інформаційної безпеки та/або кібербезпеки на об'єктах інформаційної діяльності та критичної інфраструктури.

Забезпечувати безперервність бізнес/операційних процесів, а також виявляти уразливості інформаційних систем та ресурсів, аналізувати та оцінювати ризики для інформаційної безпеки та/або кібербезпеки організації.

Обирати, аналізувати і розробляти придатні типові аналітичні, розрахункові та експериментальні методи кіберзахисту, розробляти, реалізовувати та супроводжувати проекти з захисту інформації у кіберпросторі, інноваційної діяльності та захисту інтелектуальної власності.

1. Теоретичний матеріал

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.

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.

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.

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).

На рисунку 2.1 зображено розташування розглянутих елементів методу SADT при описі бізнес-процесів.

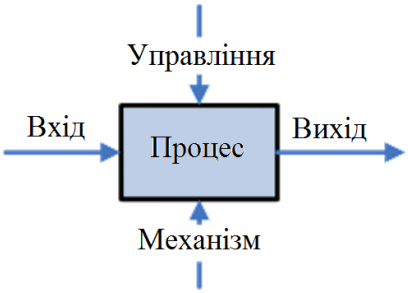


Рисунок 2.1 – Опис бізнес-процесу методом SADT

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 2.2 продемонстровано результат опису поставленої задачі.

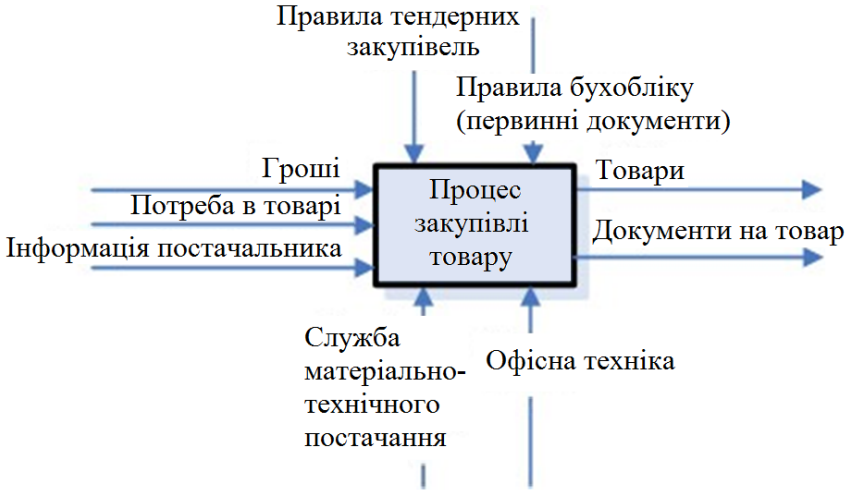


Рисунок 2.2 – Опис процесу закупівлі товару методом SADT

2. Завдання до лабораторних робіт № 2

Оберіть задачу з області кібербезпеки, захисту інформації або іншої діяльності. Сформулювати обрану задачу. Описати головний процес та його підпроцеси, ціль, точку зору, результати, ресурси, власника, виконавців, управління та механізми. Оформити описану задачу згідно методології SADT (тільки діаграма верхнього рівня)

Лабораторні роботи № 3. Дослідження структури та умов використання методології IDEF0

Мета роботи:

Розглянути визначення бізнес-процесу та спосіб його опису за допомогою методології IDEF0. Навчитися розробляти задачі у вигляді бізнес-процесу, описаного за методологією IDEF0. Моделювання діаграм A-0, A0 та An.

Результати навчання:

Досліджувати, розробляти і супроводжувати системи та засоби інформаційної безпеки та/або кібербезпеки на об’єктах інформаційної діяльності та критичної інфраструктури.

Забезпечувати безперервність бізнес/операційних процесів, а також виявляти уразливості інформаційних систем та ресурсів, аналізувати та оцінювати ризики для інформаційної безпеки та/або кібербезпеки організації.

Обирати, аналізувати і розробляти придатні типові аналітичні, розрахункові та експериментальні методи кіберзахисту, розробляти, реалізовувати та супроводжувати проекти з захисту інформації у кіберпросторі, інноваційної діяльності та захисту інтелектуальної власності.

1. Теоретичний матеріал

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.

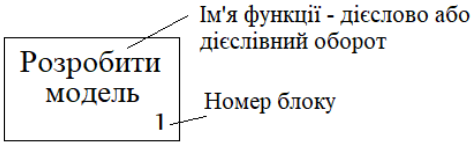
Стосовно до вже існуючих систем IDEF0 може бути використана для аналізу функцій, виконуваних системою, і відображення механізмів, за допомогою яких ці функції виконуються.

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.

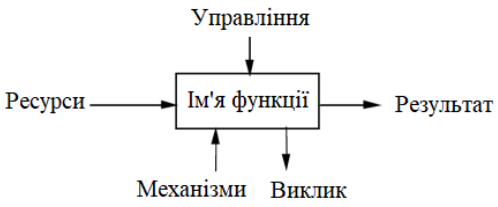


Стрілки являють дані або матеріальні об'єкти, пов'язані з функціями. Стрілки не уявляють потік або послідовність подій. Вони показують, які дані або матеріальні об'єкти мають надійти на вхід функції для того, щоб ця функція могла виконуватися.

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



Приклад діаграми IDEF0 для банківського завдання продемонстровано на рисунку 3.1.

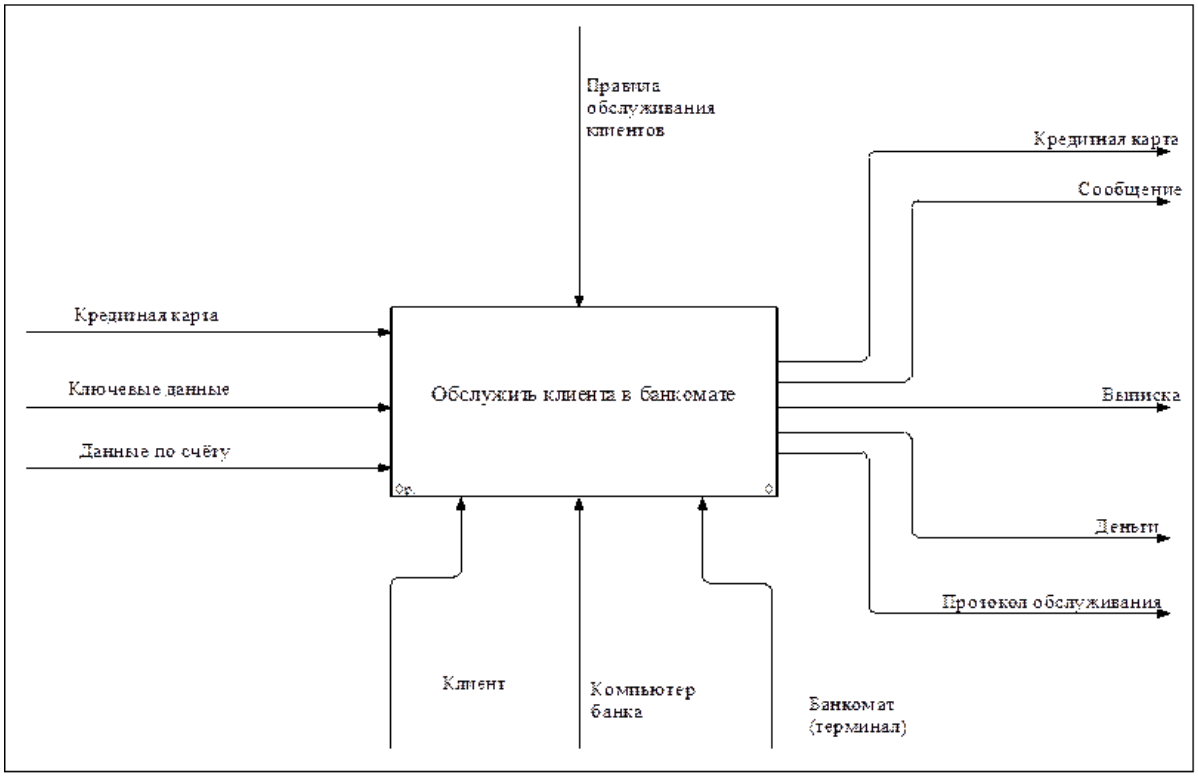


Рисунок 3.1 – Приклад процесу «Обслужити клієнта в банкоматі»

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.

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 3.2).

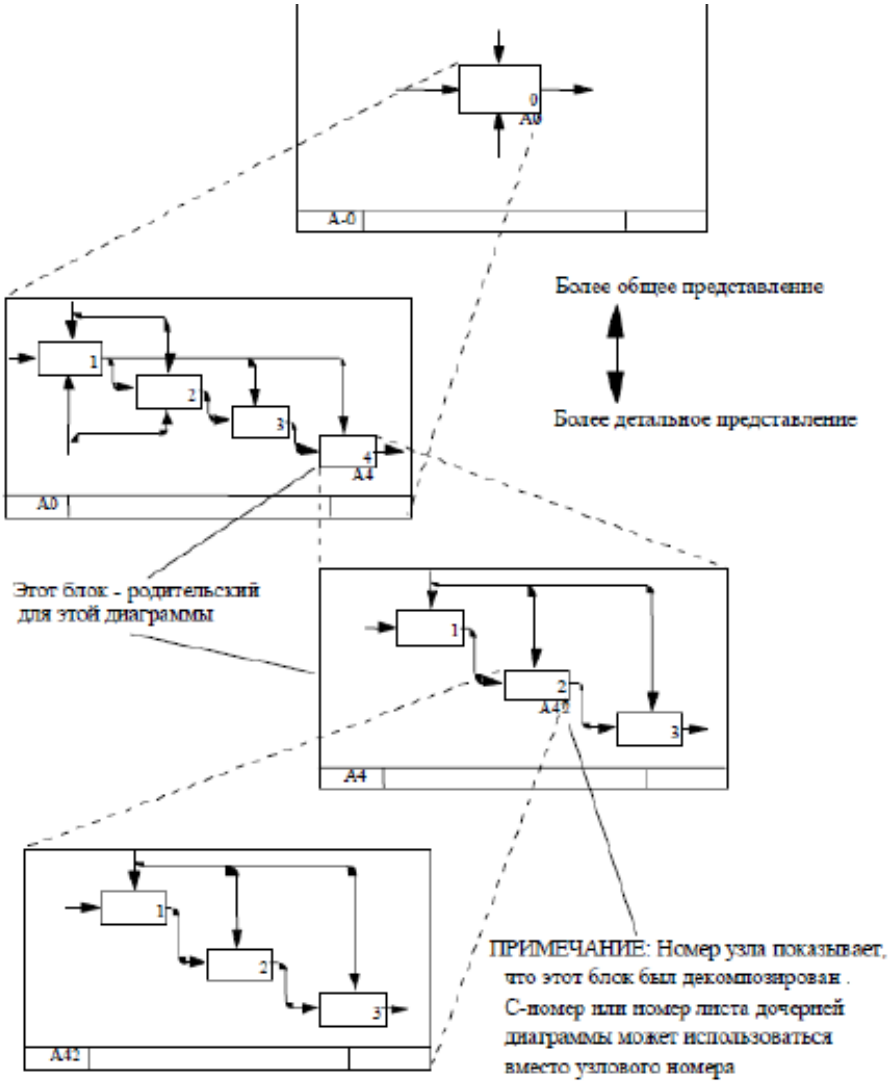
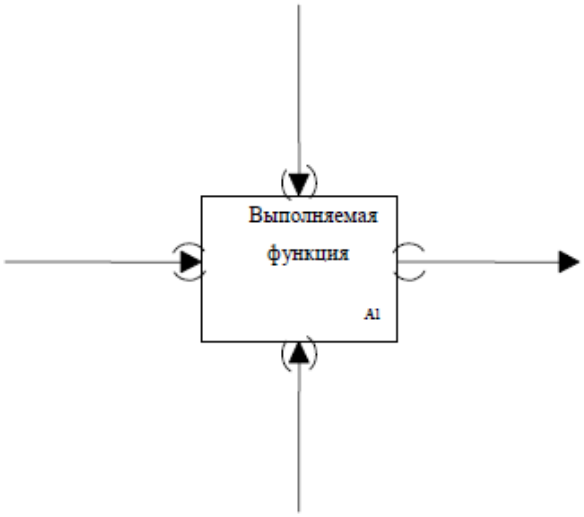


Рисунок 3.2 – Приклад декомпозиції

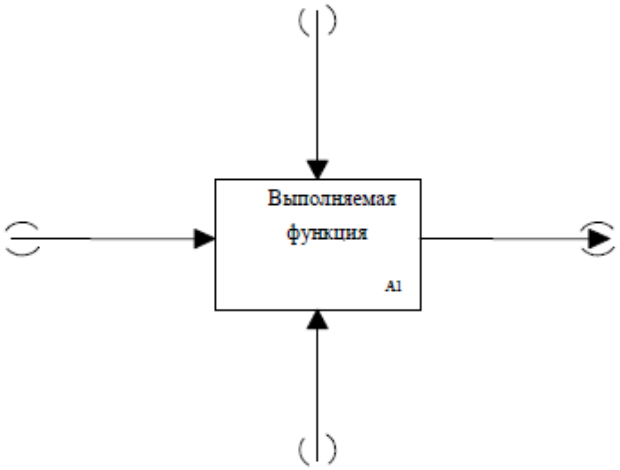
Ця діаграма, звана *діаграмою-нащадком*, описує все, пов'язане з цим блоком і його дугами, і не описує нічого поза цією межею. Декомпозіруємий блок називається *батьківським блоком*, а що містить його діаграма – *батьківської діаграмою*.

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, вміщена в тунель там, де вона приєднується до блоку означає, що дані, виражені цією 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.



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.

Приклад контекстних діаграм процесу «Написання статті» продемонстровані на рисунку 3.3 та 3.4.

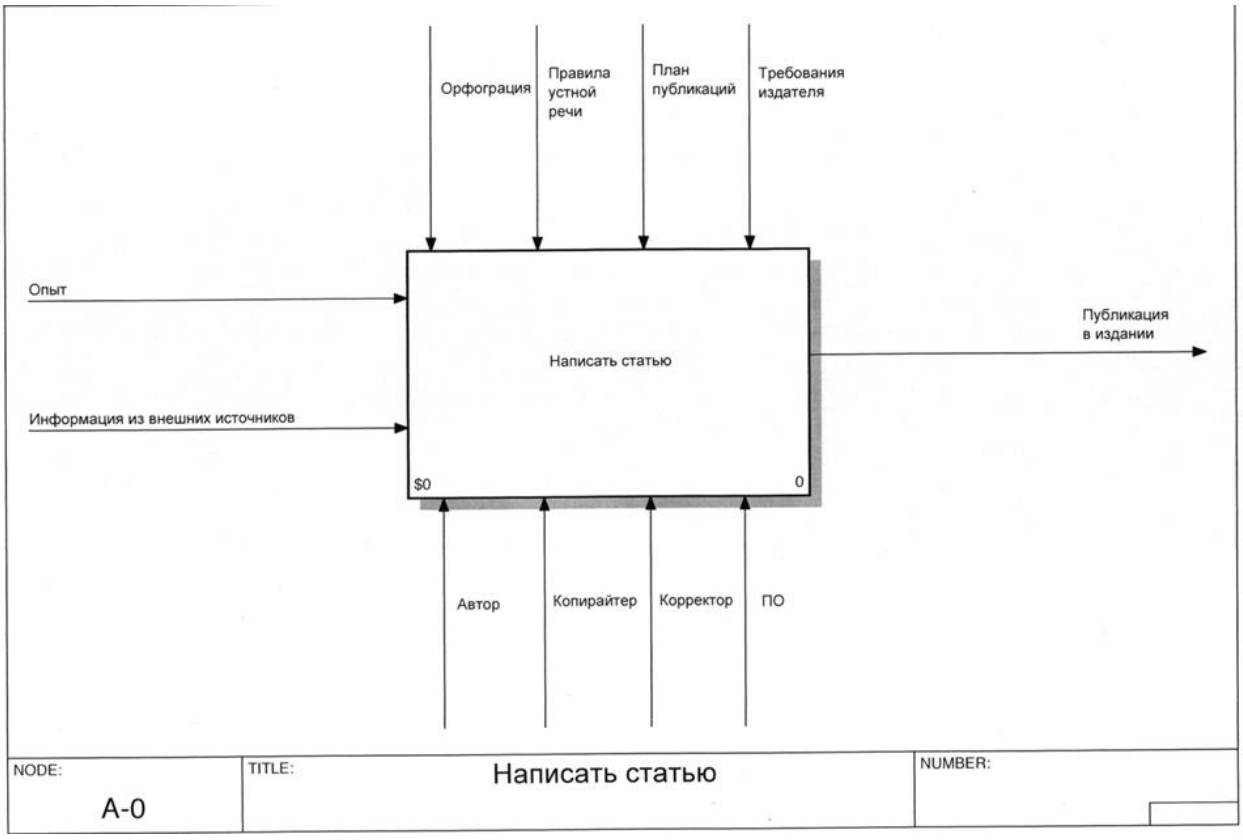


Рисунок 3.3 – Приклад контекстної діаграми А-0 процесу «Написання статті»

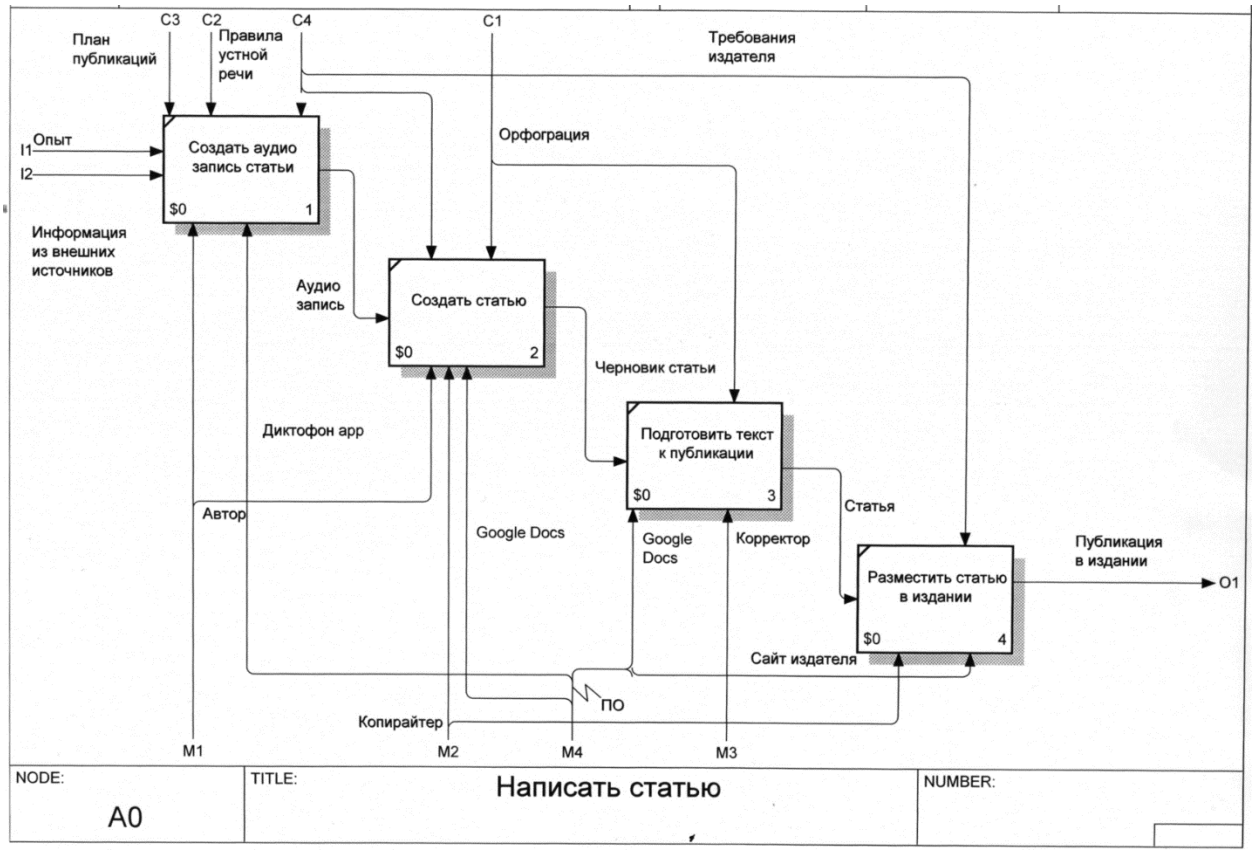


Рисунок 3.4 – Приклад контекстної діаграми А0 процесу «Написання статті»

2. Завдання до лабораторних робіт № 3

Оберіть задачу з області кібербезпеки, захисту інформації або іншої діяльності. Сформулювати обрану задачу. Описати головний процес та перелік підпроцесів, ціль, точку зору, результати, ресурси, власника, виконавців, управління та механізми. Оформити описану задачу згідно методології IDEF0 (діаграми A-0, A0, An).

Семестровий модуль 2

Лабораторна робота № 4. Дослідження структури та умов використання методології IDEF3

Мета роботи:

Розглянути визначення бізнес-процесу та спосіб його опису за допомогою методології IDEF3. Навчитися розробляти задачі у вигляді бізнес-процесу, описаного за методологією IDEF3. Моделювання діаграм PFDD та OSTN.

Результати навчання:

Досліджувати, розробляти і супроводжувати системи та засоби інформаційної безпеки та/або кібербезпеки на об’єктах інформаційної діяльності та критичної інфраструктури.

Забезпечувати безперервність бізнес/операційних процесів, а також виявляти уразливості інформаційних систем та ресурсів, аналізувати та оцінювати ризики для інформаційної безпеки та/або кібербезпеки організації.

Обирати, аналізувати і розробляти придатні типові аналітичні, розрахункові та експериментальні методи кіберзахисту, розробляти, реалізовувати та супроводжувати проекти з захисту інформації у кіберпросторі, інноваційної діяльності та захисту інтелектуальної власності.

1. Теоретичний матеріал

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.

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

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.

Розглянемо елементи діаграми опису послідовності етапів процесу (PFDD). Основними елементами IDEF3-моделі є (рисунок 4.1):

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

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.

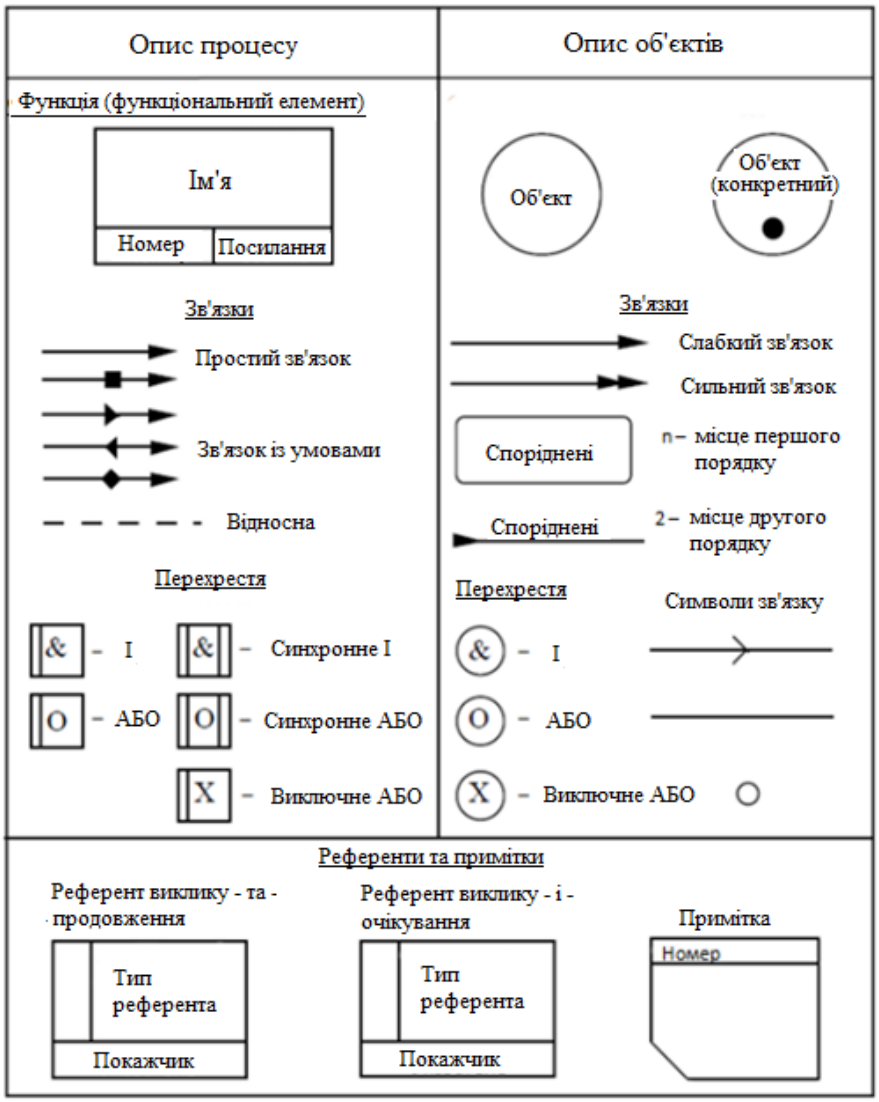
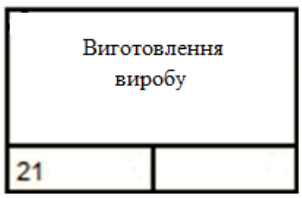


Рисунок 4.1 – Позначення основних елементів IDEF3-моделі

Приклад функції:



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.

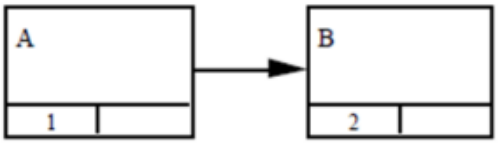
У переважній більшості випадків використовуються зв'язки, що відображають просте тимчасове відношення між блоками.

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.

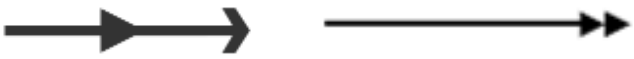
Приклад позначення простої черговості:



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.

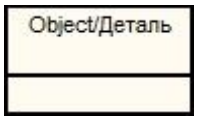
Усі зв'язки, які використовуються в схемі, в обов'язковому порядку маркуються і нумеруються.

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:

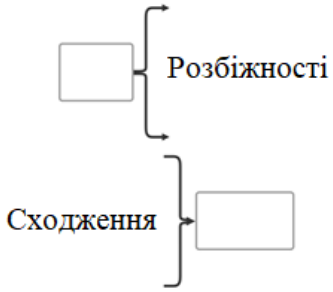


Приклад використання:

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.

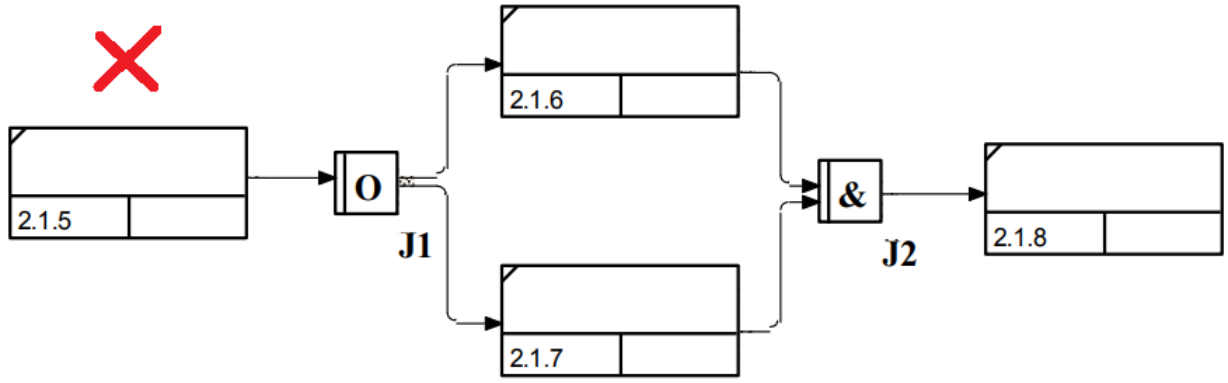
Повна синхронізація можлива тільки в разі автоматичних операцій. У бізнес-процесах беруть участь люди, тому хтось може закінчити роботу раніше, а хтось кілька затягне

процес. Але в разі синхронного вузла, передати в нього результати роботи можна буде тільки одночасно, коли всі паралельні процеси будуть завершені.

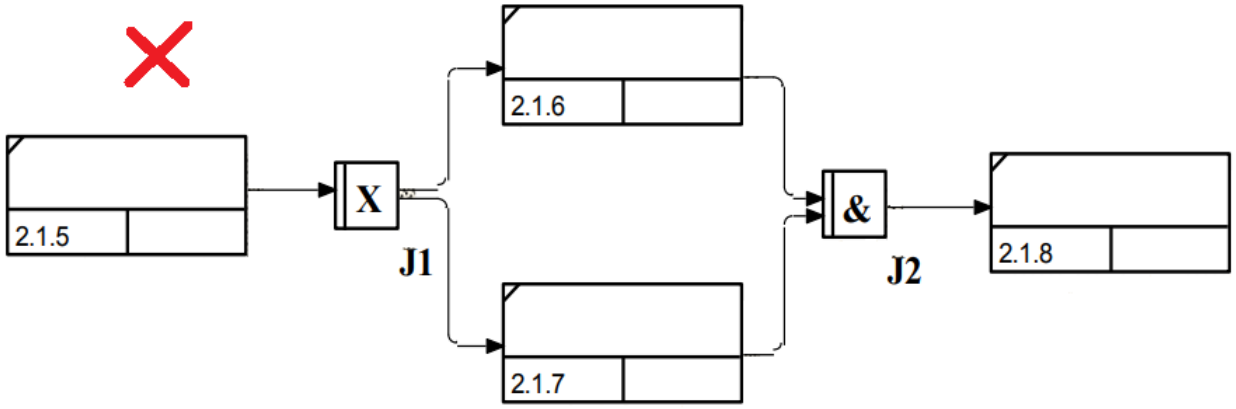
Всі з'єднання на діаграмі повинні бути парними. Однак при цьому типи з'єднань не зобов'язані збігатися. На діаграмі з'єднання зазвичай позначаються літерою «J» та цифрою.

Правила створення перехресть :

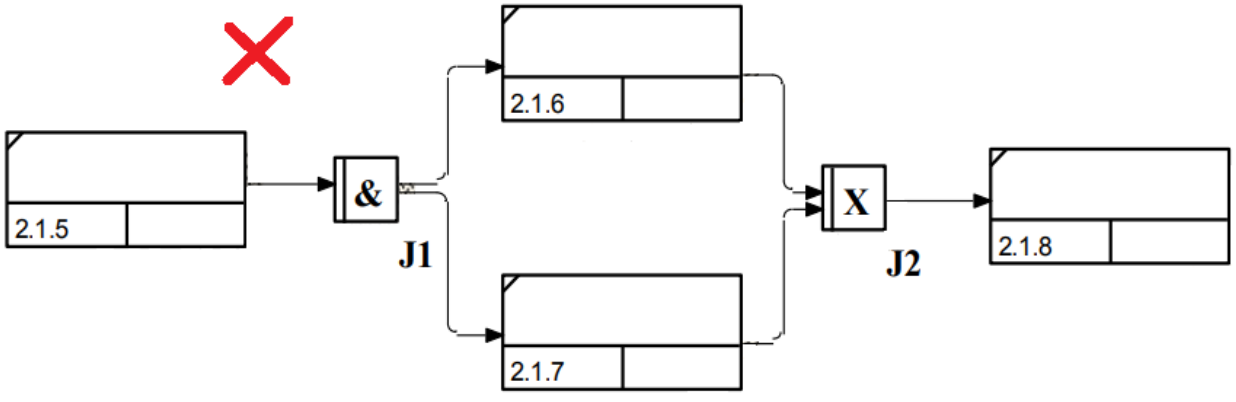
– Кожному перехрестю для злиття повинен передувати перехрестя для розгалуження.
– Перехрестя для злиття «І» не може слідувати за перехрестям для розгалуження типу синхронного або асинхронного «АБО».



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



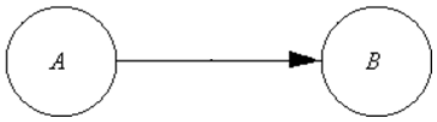
– Перехрестя для злиття типу виняткового «АБО» не може слідувати за перехрестям для розгалуження типу «І»



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

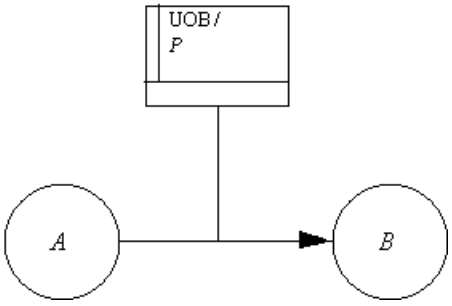
Якщо діаграми PFDD технологічний процес «З точки зору спостерігача», то інший клас діаграм IDEF3 *Діаграми переходу стану об'єкта* (OSTN) дозволяє розглядати той же самий процес «З точки зору об'єкта».

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.

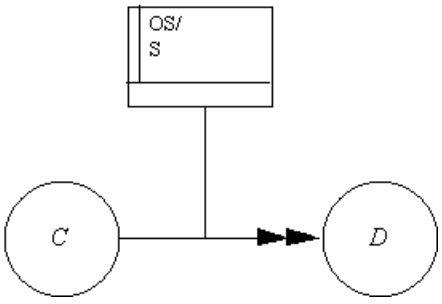
Наприклад:



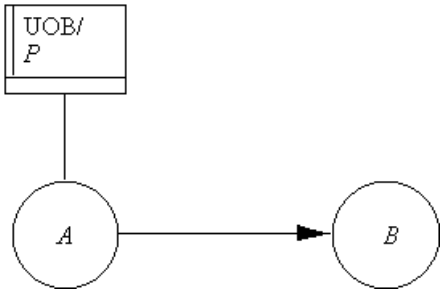
Існують декілька типів таких об'єктів:

– 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).

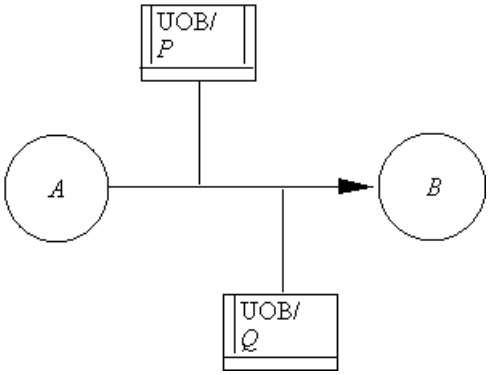
Приклад позначення «Викликай та продовжуй»



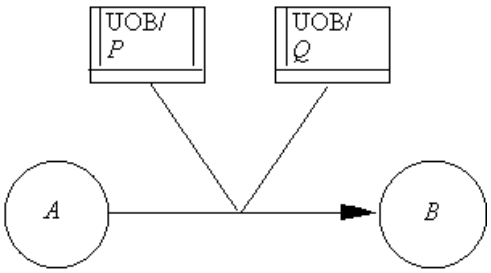
Існують референти, прикріплені до об'єктних станів. У даній ситуації досить часто використовується «утримання» об'єкта в даному стані; наприклад, в процесі заморожування дана речовина може підтримуватися в твердому стані. Ситуації цього типу можна представити конструкцією:



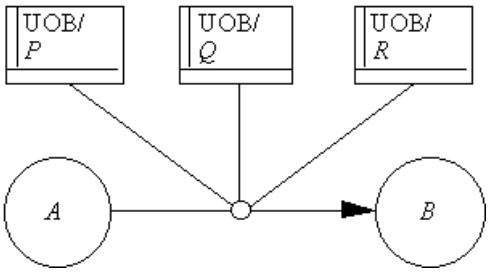
Схематики об'єктів з безліччю референтів. При переході з одного стану в інший часто може спостерігатися більш складний хід подій, в порівнянні з тим ходом подій, який може бути визначений одним референтом. Можна припускати, що деталі такого перебігу подій можуть бути забезпечені окремою схематики процесів.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

Розглянемо приклад діаграми PFDD (рисунок 4.2) для завдання «Забарвлення деталі» та діаграму OSTN (рисунок 4.3) для цього ж завдання.

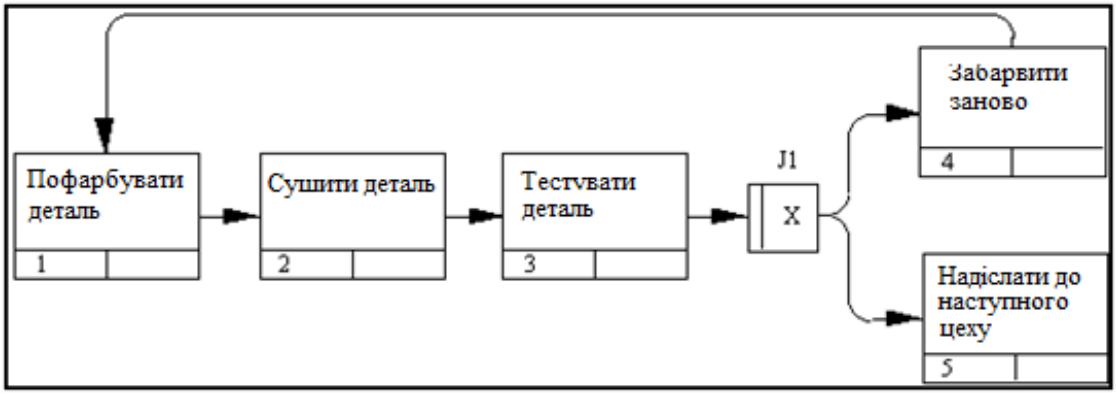


Рисунок 4.2 – Приклад діаграми PFDD

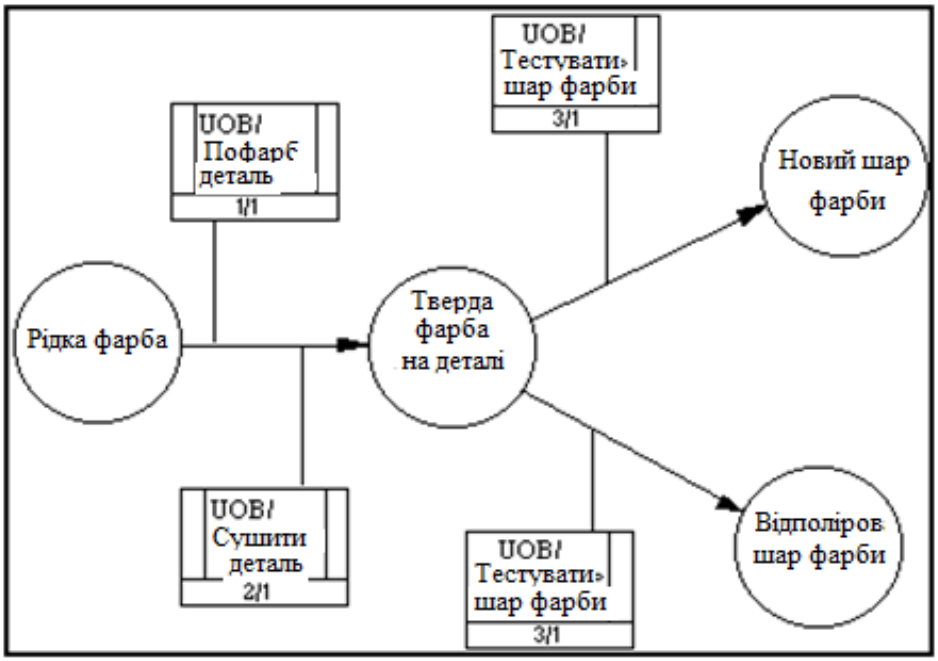


Рисунок 4.3 – Приклад діаграми OSTN

2. Завдання до лабораторної роботи №4

Оберіть з попередніх лабораторних занять одну або декілька підпроцесів. Створити для обраних процесів діаграму опису послідовності етапів процесу (PFDD) та діаграму переходу стану об'єкта (OSTN) методології IDEF3.

Лабораторна робота № 5. Дослідження структури та умов використання методології IDEF1x

Мета роботи:

Розглянути визначення бізнес-процесу та спосіб його опису за допомогою методології IDEF1x. Навчитися розробляти задачі у вигляді бізнес-процесу, описаного за методологією IDEF1x.

Результати навчання:

Досліджувати, розробляти і супроводжувати системи та засоби інформаційної безпеки та/або кібербезпеки на об’єктах інформаційної діяльності та критичної інфраструктури.

Забезпечувати безперервність бізнес/операційних процесів, а також виявляти уразливості інформаційних систем та ресурсів, аналізувати та оцінювати ризики для інформаційної безпеки та/або кібербезпеки організації.

Обирати, аналізувати і розробляти придатні типові аналітичні, розрахункові та експериментальні методи кіберзахисту, розробляти, реалізовувати та супроводжувати проекти з захисту інформації у кіберпросторі, інноваційної діяльності та захисту інтелектуальної власності.

1. Теоретичний матеріал

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.

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.

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.

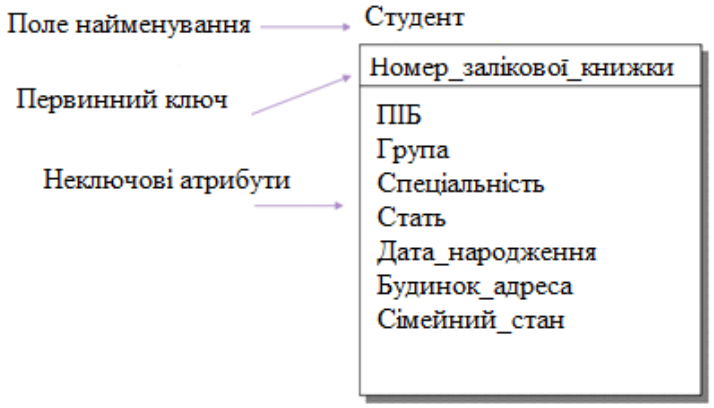
Атрибут представляє тип характеристик або властивостей, асоційованих з безліччю реальних або абстрактних об'єктів (людей, об'єктів, місць, подій, станів, ідей, пар предметів тощо).

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

Сутність повинна мати атрибут 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.

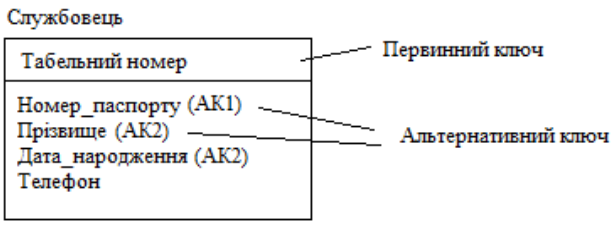
Успадковані передаються від сутності – батька при визначенні зв'язку.

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.

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.

Специфіческое 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 Р, СКЛАДАЄТЬСЯ_З – як ім'я відносини.

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*.

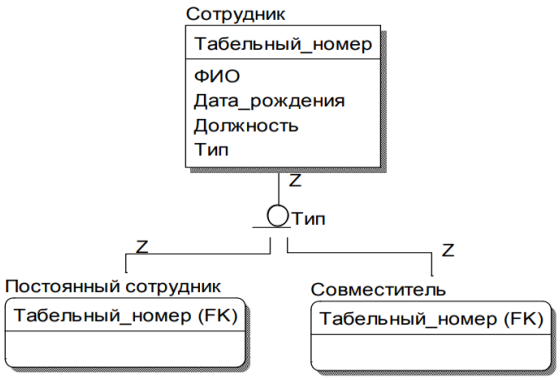
Відношення категоризації зображується лінією, що веде з загальної суті до підкресленому колу. Окремі лінії ведуть від підкресленого кола до кожної з сутностей-категорій.

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.

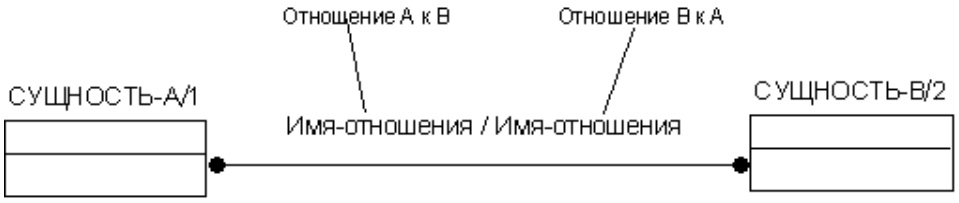
Наприклад:



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 (рисунок 5.1):

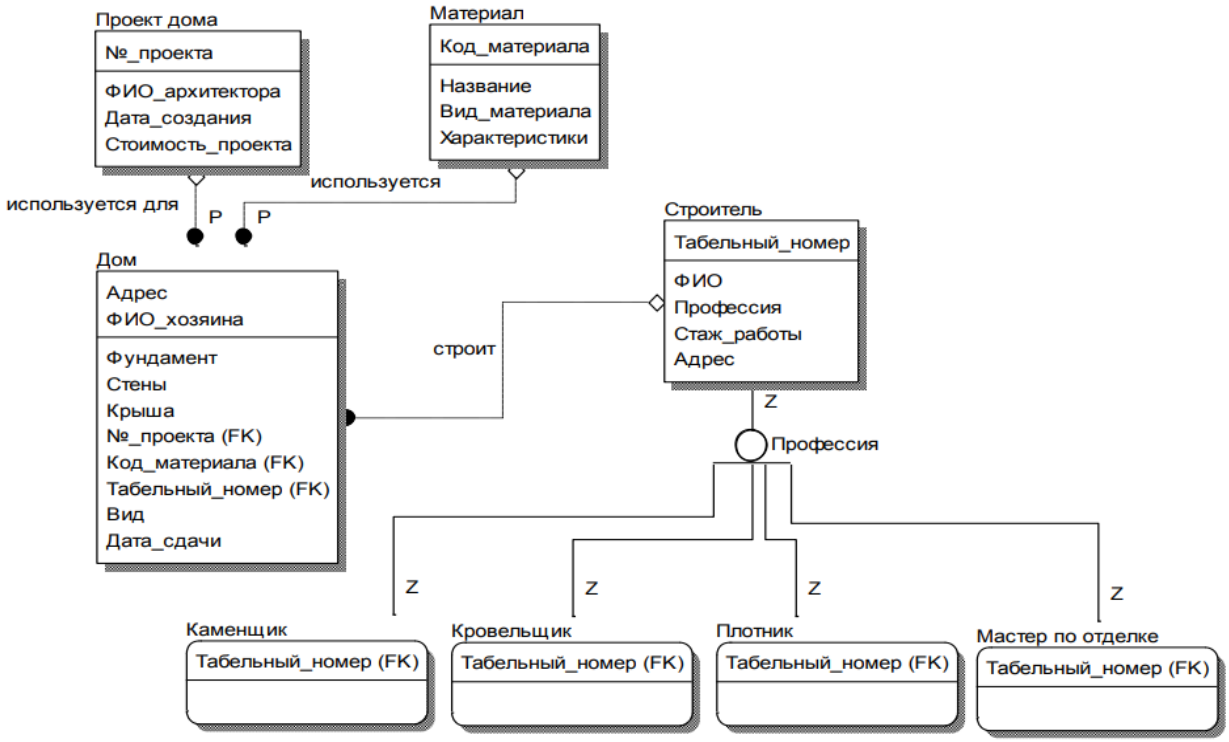


Рисунок 5.1 – Інформаційна модель процесу спорудження будиночка

2. Завдання до лабораторної роботи №5

Оберіть задачу з області кібербезпеки, захисту інформації або іншої діяльності. Сформулювати обрану задачу. Описати перелік сутностей, їх атрибути, ключі та зв’язки між ними для обраної задачі. Оформити описану задачу згідно методології IDEF1x. Оформити описану задачу згідно методології ERD. Порівняти отримані результати.

Лабораторна робота № 6. Дослідження структури та умов використання методології UML. Модель класів

Мета роботи:

Розглянути визначення бізнес-процесу та спосіб його опису за допомогою методології UML. Навчитися розробляти задачі у вигляді бізнес-процесу, описаного за методологією UML. Розглядається структурне моделювання, а саме модель та діаграма класів.

Результати навчання:

Досліджувати, розробляти і супроводжувати системи та засоби інформаційної безпеки та/або кібербезпеки на об’єктах інформаційної діяльності та критичної інфраструктури.

Забезпечувати безперервність бізнес/операційних процесів, а також виявляти уразливості інформаційних систем та ресурсів, аналізувати та оцінювати ризики для інформаційної безпеки та/або кібербезпеки організації.

Обирати, аналізувати і розробляти придатні типові аналітичні, розрахункові та експериментальні методи кіберзахисту, розробляти, реалізовувати та супроводжувати проекти з захисту інформації у кіберпросторі, інноваційної діяльності та захисту інтелектуальної власності.

1. Теоретичний матеріал

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).

Один клас описується у вигляді прямокутника. Прямокутник поділено на 3 секції:

– верхня секція містить назву класу;

– середня секція описує стан, тому містить перелік полів (атрибутів) класу, імена та типи даних, разом із указанням області видимості;
– нижня секція описує поведінку, тому містить містить перелік методів (операцій) – імена методів, їх тип (статичні чи нестатичні), тип повернення, перелік вхідних параметрів тощо; також в цій секції указують конструктори, які прописані в явному виді у коді програми.

Наприклад:

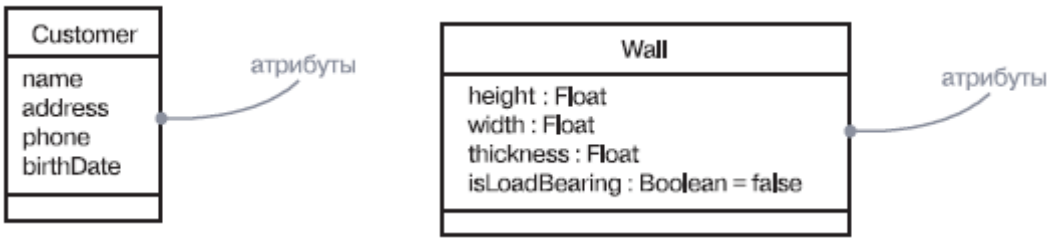


Рисунок 6.1 – Приклади класів

У UML атрибути показуються щонайменше назвою, також може бути показано їх тип, початкове значення і інші властивості. Крім того, атрибути може бути показано з областю видимості атрибута:

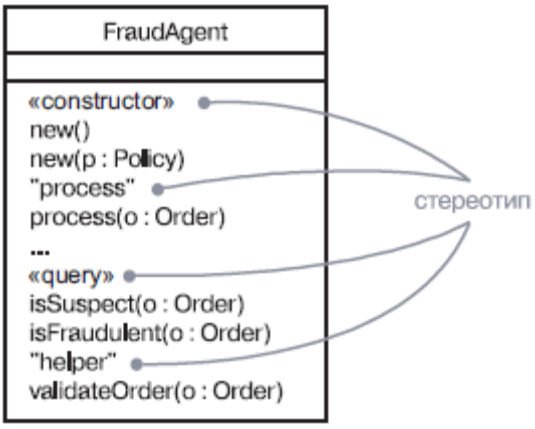
– + відповідає публічним (public) атрибутам
– # відповідає захищеним (protected) атрибутам
– - відповідає приватним (private) атрибутам

Операції (методи) також показуються принаймні назвою, крім того, може бути показано їх параметри і типи значень, які буде повернуто. Операції, як і атрибути, може бути показано з областю видимості:

– + відповідає публічним (public) операціям
– # відповідає захищеним (protected) операціям
– - відповідає приватним (private) операціям

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.

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

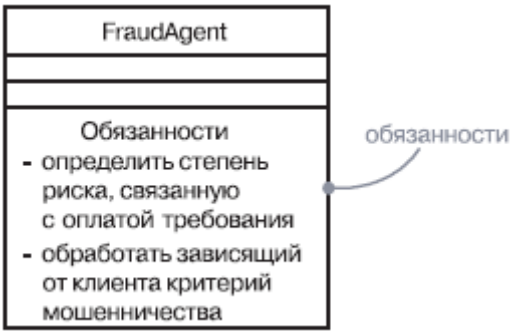


Обов'язок (responsibility) – це угода або зобов'язання класу. Коли ви створюєте клас, висувається припущення, що всі його об'єкти характеризуються однаковим станом і однаковою поведінкою.

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.

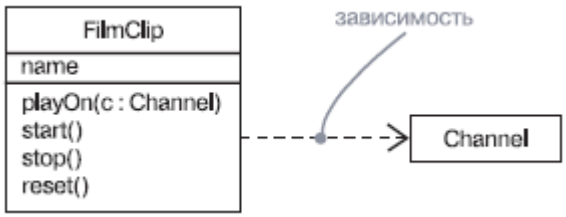
Залежність являє собою зв'язок використання. Наприклад, труби залежать від водонагрівача для підігріву води, яка по ним передається.

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.



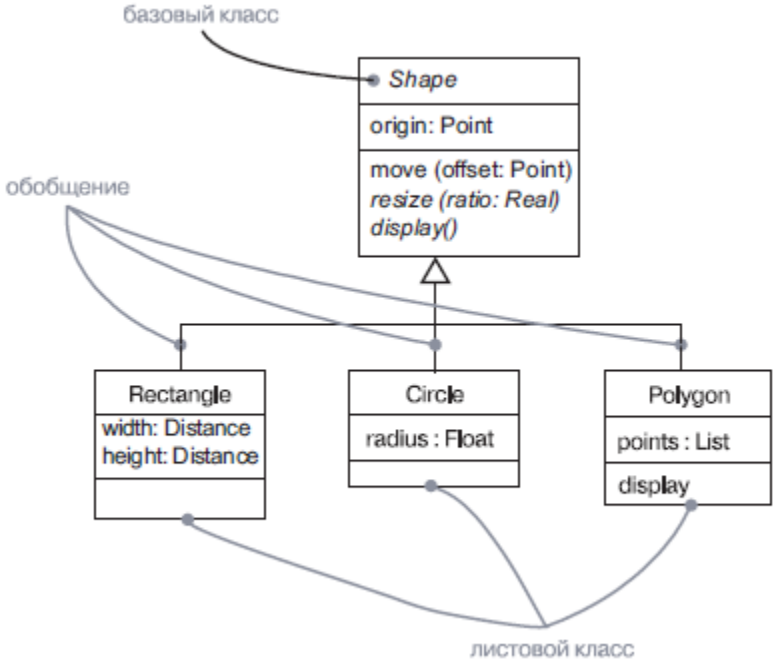
Вибирається залежність, коли потрібно показати, що одна сутність використовує іншу. Найчастіше цей тип зв'язку застосовується для того, щоб показати, що один клас використовує операції іншого класу (або використовує змінні чи аргументи типу іншого класу).

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.

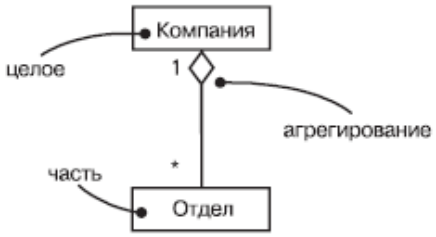
Множинність може бути визначена як одиниця (1), нуль або один (0..1), багато (0 .. *), один або кілька (1 .. *). Можна задавати діапазон цілих значень, наприклад 2..5, або встановлювати точне число, наприклад 3 (еквівалент записи 3..3).

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.

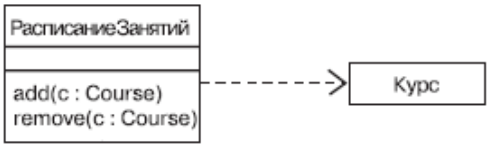
Можна також змоделювати зв'язок «ціле-частина», в якій один клас представляє велику сутність (ціле), що містить в собі більш дрібні (частини). Цей тип зв'язків, заснованих на відносинах володіння, називається *агрегацією* і має на увазі, що об'єкт-ціле володіє об'єктами-частинами.

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:



Приклад діаграми класів (рисунок 6.2).

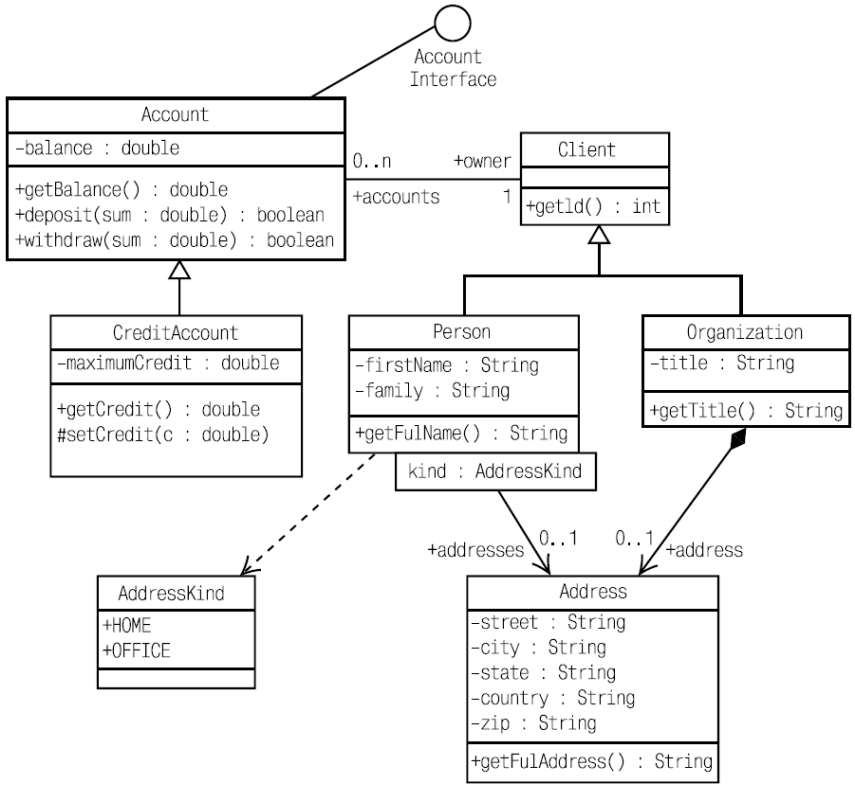


Рисунок 6.1 – Приклад діаграми класів

2. Завдання до лабораторної роботи №6

Оберіть задачу з області кібербезпеки, захисту інформації або іншої діяльності. Сформулювати обрану задачу. Описати перелік класів, їх атрибути, операції та зв’язки між класами для обраної задачі. Оформити описану задачу у вигляді діаграми класів методології UML.

Лабораторна робота № 7. Дослідження структури та умов використання методології UML. Моделі стану та взаємодії

Мета роботи:

Розглянути визначення бізнес-процесу та спосіб його опису за допомогою методології UML. Навчитися розробляти задачі у вигляді бізнес-процесу, описаного за методологією UML. Розглядається моделювання поведінки, а саме модель та діаграма взаємодії та станів.

Результати навчання:

Досліджувати, розробляти і супроводжувати системи та засоби інформаційної безпеки та/або кібербезпеки на об’єктах інформаційної діяльності та критичної інфраструктури.

Забезпечувати безперервність бізнес/операційних процесів, а також виявляти уразливості інформаційних систем та ресурсів, аналізувати та оцінювати ризики для інформаційної безпеки та/або кібербезпеки організації.

Обирати, аналізувати і розробляти придатні типові аналітичні, розрахункові та експериментальні методи кіберзахисту, розробляти, реалізовувати та супроводжувати проекти з захисту інформації у кіберпросторі, інноваційної діяльності та захисту інтелектуальної власності.

1. Теоретичний матеріал

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.

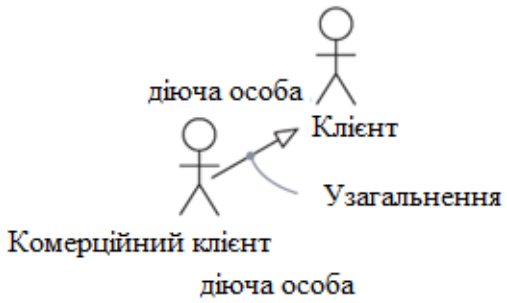
Варіанти використання та діючі особи в UML зображуються, як показано на рисунку 7.1.

Рисунок 7.1 – Приклад зображення варіанта використання та діючій особи

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.

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.

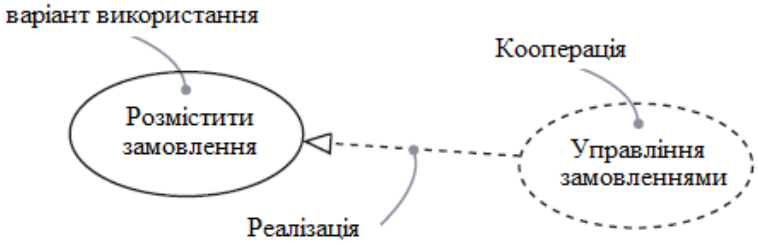
Діючі особи зображуються у вигляді людських фігурок. Можна вводити загальні типи діючих осіб, такі як Customer, і спеціалізувати їх (наприклад, створити різновид CommercialCustomer – комерційний клієнт), визначивши зв'язок узагальнення.



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.

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.

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.

2. Завдання до лабораторної роботи №7

Для одного з описаних класів або обраної задачі з лабораторного заняття №6 створити діаграму варіантів використання методології UML.

Лабораторна робота № 8. Дослідження структури та умов використання методології BPMN

Мета роботи:

Розглянути визначення бізнес-процесу та спосіб його опису за допомогою методології BPMN. Навчитися розробляти задачі у вигляді бізнес-процесу, описаного за методологією BPMN.

Результати навчання:

Досліджувати, розробляти і супроводжувати системи та засоби інформаційної безпеки та/або кібербезпеки на об’єктах інформаційної діяльності та критичної інфраструктури.

Забезпечувати безперервність бізнес/операційних процесів, а також виявляти уразливості інформаційних систем та ресурсів, аналізувати та оцінювати ризики для інформаційної безпеки та/або кібербезпеки організації.

Обирати, аналізувати і розробляти придатні типові аналітичні, розрахункові та експериментальні методи кіберзахисту, розробляти, реалізовувати та супроводжувати проекти з захисту інформації у кіберпросторі, інноваційної діяльності та захисту інтелектуальної власності.

1. Теоретичний матеріал

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.

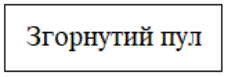
Весь бізнес-процес складається з *Пулів* (Pool) – це сукупності операцій та осіб, які ці операції виконують.

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:

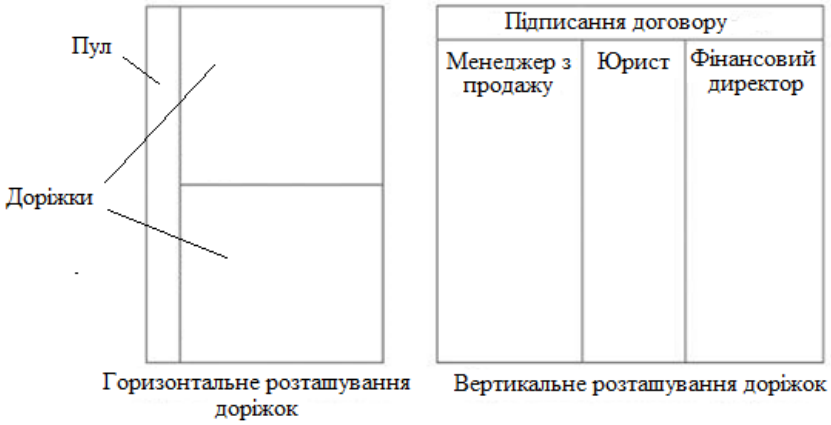
Згорнутий пул – елемент, що позначає зовнішній (по відношенню до поточної діаграмі) процес або зовнішнє посилання. Усередині блоку поміщається найменування зовнішнього процесу або зовнішнього посилання.



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.

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:

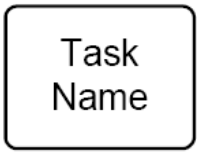


У кожній доріжці розташовуються дії, що виконуються одним виконавцем.

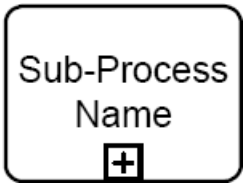
Під *Дією* розуміється одиниця роботи, що виконується в ході виконання бізнес-процесу. Дії можуть бути як елементарними (завдання / task), так і складовими (підпроцес / sub-process).

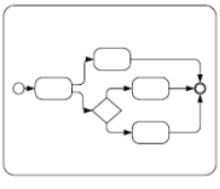
BPMN передбачає наступні графічні відображення для основних типів дій:

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) 

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.

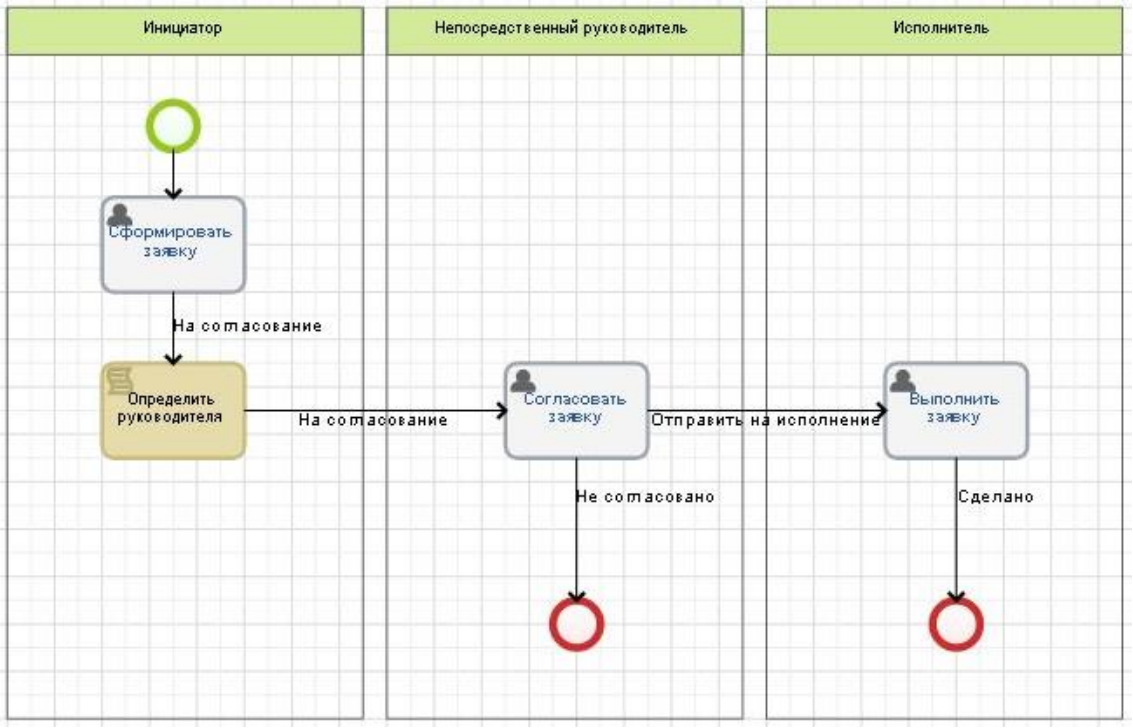
Події-сигнали (signal events) розсилають і приймають сигнали між декількома процесами. Один сигнал може оброблятися декількома одержувачами. Таким чином, події-сигнали дозволяють реалізувати трансляцію розсилання повідомлень.

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.

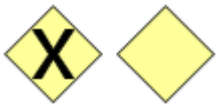
Події-останови (terminate events) призводять до негайного завершення всього бізнес процесу (у всій діаграмі).

Наприклад, продаж та відвантаження обладнання:

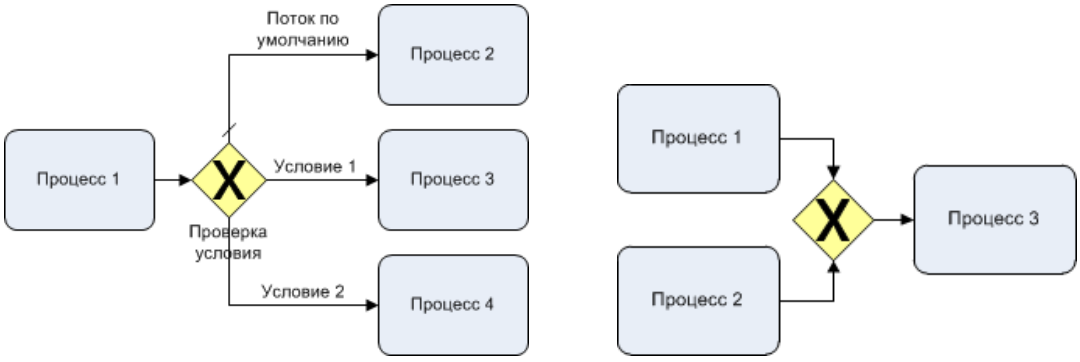


Під *шлюзами* розуміються елементи, що визначають розгалуження та злиття потоків робіт. Шлюз можна уявити як пропускний пристрій, який або пропускає потік, або ні.

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.

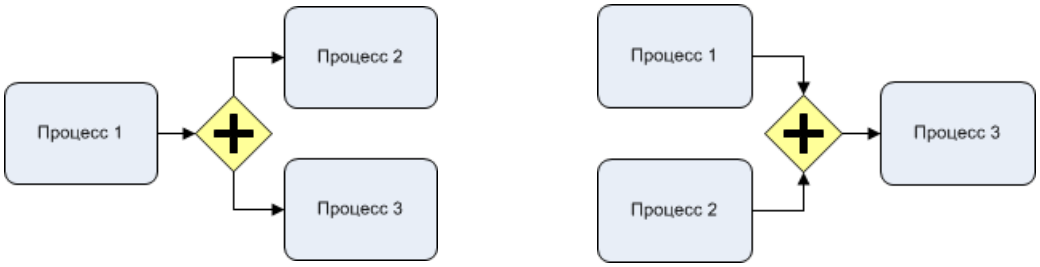


Наприклад:



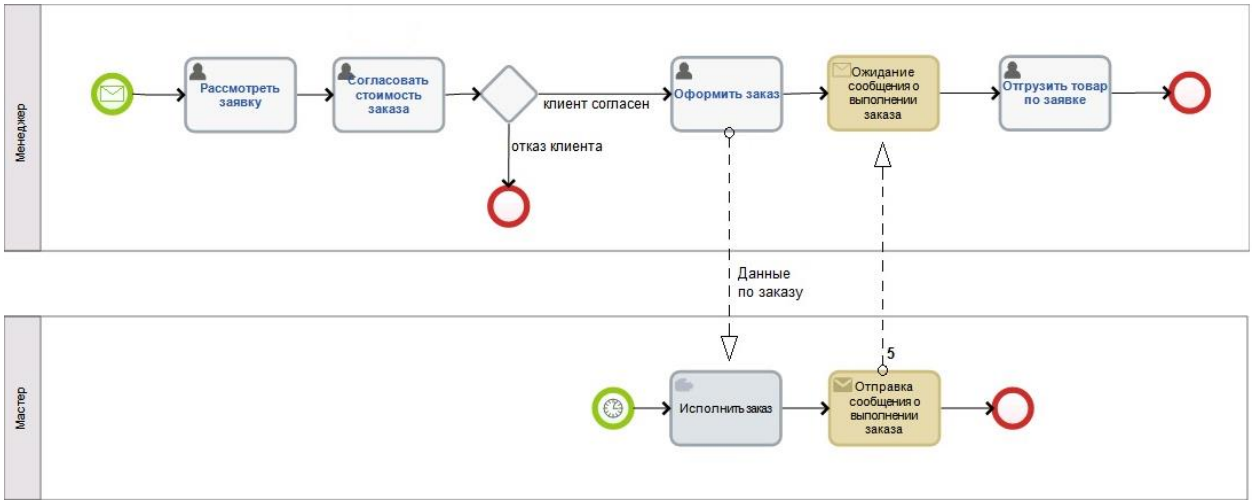
Паралельний шлюз – використовується для створення паралельних шляхів без оцінки якої б то не було умови або для потоків, що сходяться та синхронізації паралельних гілок виконання процесу.

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.

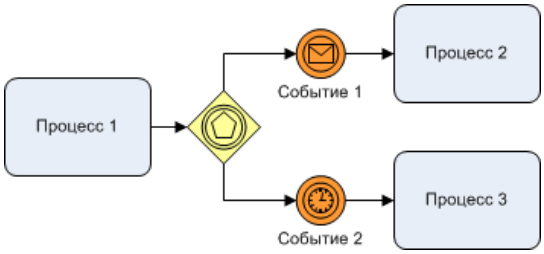
Приклад:



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.

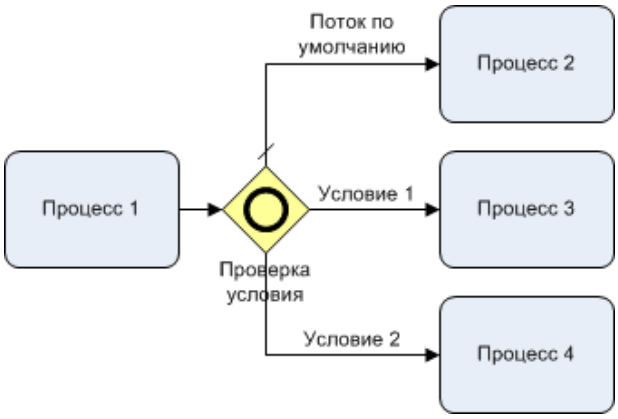
Наприклад, якщо першим виникла Подія 1, то виконається тільки Процес 2; якщо першим виникла Подія 2, то виконається тільки Процес 3.



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

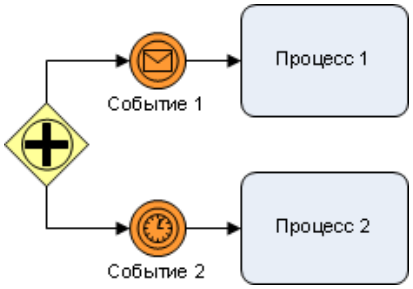
Скажімо, якщо покупцеві сподобався товар А, це спровокує один процес. Якщо покупець віддав перевагу товару Б, в силу вступить інший процес. А якщо покупець залишився незадоволений товаром А, – третій.

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.



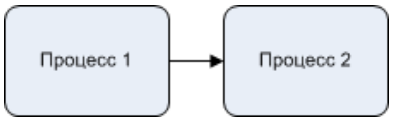
Паралельна розвилка з залежністю від події – схожа на паралельну розвилку, як видно з назви. Дозволяє декільком процесам розгортатися одночасно, тільки на відміну від паралельної розвилки, процеси залежать від конкретних подій.

Приклад позначення:



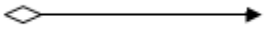
Потік – це послідовність дій, яка позначається стрілкою. Елемент «потік» показує яку дію після якого необхідно здійснити.

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

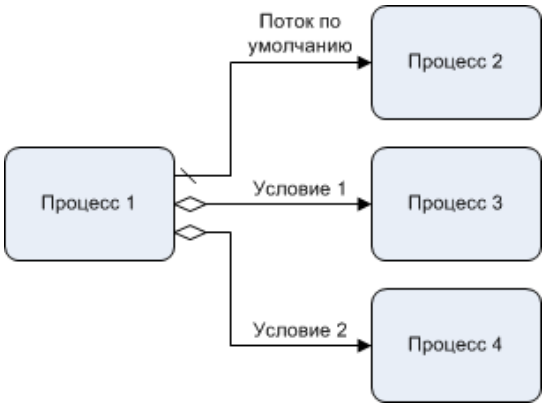


Потік управління за замовчуванням – використовується тоді, коли необхідно показати, що подальше виконання процесу буде відбуватися за певним потоком тільки якщо не виконується жодна з заданих умов.

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.



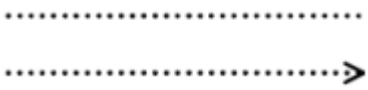
Приклад позначення:

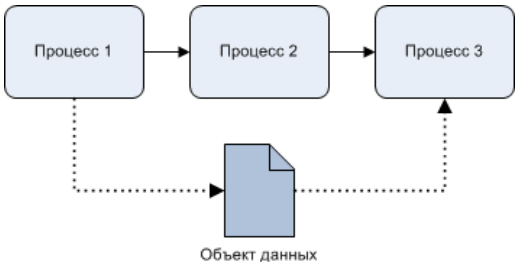


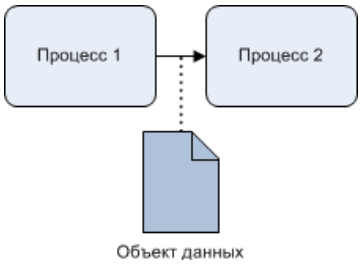
Ромбик біля основи стрілки додається, якщо умовний потік управління є вихідним від процесу. Ромбик не додають, якщо умовний потік управління є вихідним від шлюзу.

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

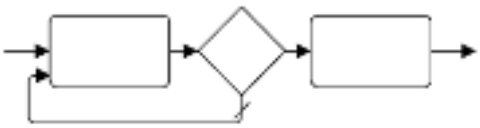
Асоціація – застосовується для візуалізації зв'язку між елементами потоку і об'єктами, які не є елементами потоку (артефактами).







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



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.

1.2 Завдання до лабораторної роботи №8

Оберіть задачу з області кібербезпеки, захисту інформації або іншої діяльності. Сформулювати обрану задачу. Описати пули, ролі та доріжки. Сформулювати перелік дій, подій, потоків та шлюзів і артефактів при потребі для обраної задачі. Оформити описану задачу у вигляді діаграми методології BPMN.

Лабораторна робота № 9. Дослідження структури та умов використання методології DFD

Мета роботи:

Розглянути визначення бізнес-процесу та спосіб його опису за допомогою методології DFD. Навчитися розробляти задачі у вигляді бізнес-процесу, описаного за методологією DFD.

Результати навчання:

Досліджувати, розробляти і супроводжувати системи та засоби інформаційної безпеки та/або кібербезпеки на об’єктах інформаційної діяльності та критичної інфраструктури.

Забезпечувати безперервність бізнес/операційних процесів, а також виявляти уразливості інформаційних систем та ресурсів, аналізувати та оцінювати ризики для інформаційної безпеки та/або кібербезпеки організації.

Обирати, аналізувати і розробляти придатні типові аналітичні, розрахункові та експериментальні методи кіберзахисту, розробляти, реалізовувати та супроводжувати проекти з захисту інформації у кіберпросторі, інноваційної діяльності та захисту інтелектуальної власності.

1.1 Теоретичний матеріал

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?

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 (верхньорівнева контекстна діаграма).

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.

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.	Функція Функція	Функція	Функція	Функція
Внешняя сущность Является источником или адресатом потока данных. Отображает внешние по отношению к системе сущности.	Зовнішня сутність	Зовнішня сутність	Зовнішня сутність	Зовнішня сутність

Хранилище данных Используется для хранения данных	Сховище	Сховище	Сховище	Сховище
Поток данных Объект. Потоки данных между процессами, хранилищами, внешними сущностями.	Потік даних	Потік даних	Потік даних	Потік даних

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.

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

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

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:

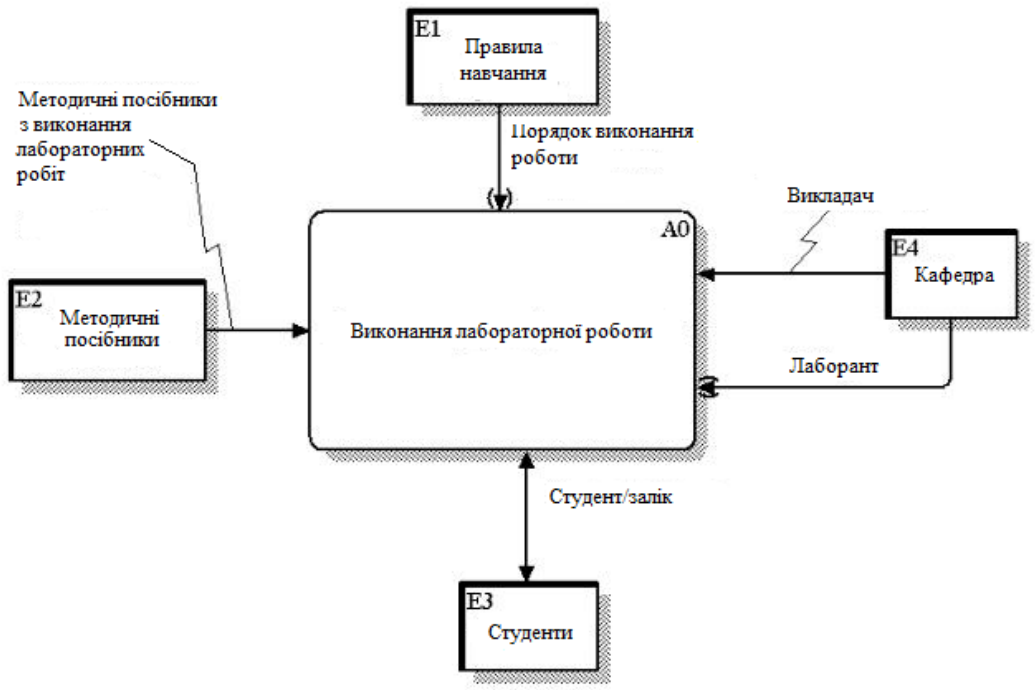


Рисунок 9.1 – Приклад процесу «Виконання лабораторної роботи»

1.2 Завдання до лабораторної роботи №9

Оберіть задачу з області кібербезпеки, захисту інформації або іншої діяльності. Сформулювати обрану задачу. Описати процеси, зовнішні сутності, потоки та сховище для обраної задачі. Оформити описану задачу у вигляді діаграми у будь-якій графічній нотації методології DFD.

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. Методологія структурного аналізу та проектування SADT. URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?id=39227
5. Нотация IDEF0. URL: http://www.businessstudio.com.ua/bp/bs/overview/notation_idef0.php
6. Business Process Modeling Notation (BPMN). Version 1.2. – Object Management Group, 2009. – 316 p.
7. Brackett, J. Applying SADT to Large System Problems / J.Brackett, C. McGowan. – SofTech Technical Paper TP059,January 1977.
8. Постіл С. Д. UML. Уніфікована мова моделювання інформаційних систем: навчальний посібник. - Ірпінь : Ун-т держ. фіск. служби України, 2019. - 321 с.
9. J. Jürjens. Secure systems development with UML. – Berlin: Springer; Heidelberg, 2005. – 309 p.