

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

МЕТОДИЧНІ ВКАЗІВКИ

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

Одеса, 2025

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

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

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

Одеса, 2025

Методичні вказівки до лабораторних робіт з дисципліни «**Бізнес-процеси в кібербезпеці**» для другого (магістерський) рівня вищої освіти за спеціальністю F5 Кібербезпека та захист інформації галузі знань F Інформаційні технології / Укл.: *Лебедєва О.Ю.* – Одеса, 2025. – 34 с.

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

ЗМІСТ

Змістовий модуль 1 5
Лабораторна робота 1. Дослідження способів опису бізнес-процесів 5
1. Теоретичний матеріал.....5
2. Завдання до лабораторних робіт 1 8
Лабораторні роботи 2. Дослідження структури та умов використання методології IDEF0 ..9
1. Теоретичний матеріал.....9
2. Завдання до лабораторних робіт 2 13
Лабораторна робота 3. Дослідження структури та умов використання методології IDEF3 15
1. Теоретичний матеріал.....15
2. Завдання до лабораторної роботи 3 23
Лабораторна робота 4. Дослідження структури та умов використання методології IDEF1x 24
1. Теоретичний матеріал.....24
2. Завдання до лабораторної роботи 4 28
Лабораторна робота 5. Дослідження структури та умов використання методології UML та діаграми класів.....29
1. Теоретичний матеріал.....29
2. Завдання до лабораторної роботи 5 39
Лабораторна робота 6. Дослідження структури та умов використання методології UML та діаграми варіантів використання 40
1. Теоретичний матеріал.....40
2. Завдання до лабораторної роботи 6 49
Лабораторна робота 7. Дослідження структури та умов використання методології UML. Діаграма станів 50
1. Теоретичний матеріал.....50
2. Завдання до лабораторної роботи 7 55
Лабораторна робота 8. Дослідження структури та умов використання методології BPMN 56
1. Теоретичний матеріал.....56
2 Завдання до лабораторної роботи 8 64
Лабораторна робота 9. Дослідження структури та умов використання методології DFD...65
1.1 Теоретичний матеріал.....65
2 Завдання до лабораторної роботи 9 67
Лабораторна робота 10. Дослідження структури та умов використання методології EPC .68
1. Теоретичний матеріал.....68
2. Завдання до лабораторної роботи 10 76
Література.....77

Змістовий модуль 1
Лабораторна робота 1. Дослідження способів опису бізнес-процесів

Мета роботи:

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

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).

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

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

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

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

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 1.1 зображено розташування розглянутих елементів методу SADT при описі бізнес-процесів.

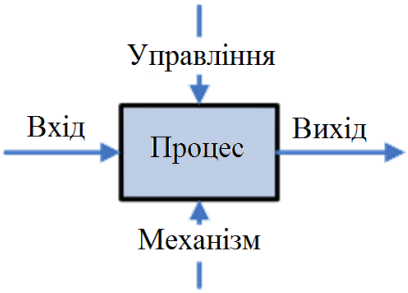


Рисунок 1.1 – Опис бізнес-процесу методом SADT

Розглянемо приклад схеми опису процесу «Здобувач здає лабораторну роботу викладачу»:

Вхід: необхідність здати лабораторну роботу, знання, завдання.

Вихід: лабораторна робота здана, оцінка.

Управління: правила ВУЗ до здачі лабораторних робіт, довідкова література по темі роботи.

Механізми: викладач, комп’ютер, програмне забезпечення.

На рисунку 1.2 продемонстровано результат опису поставленої задачі.

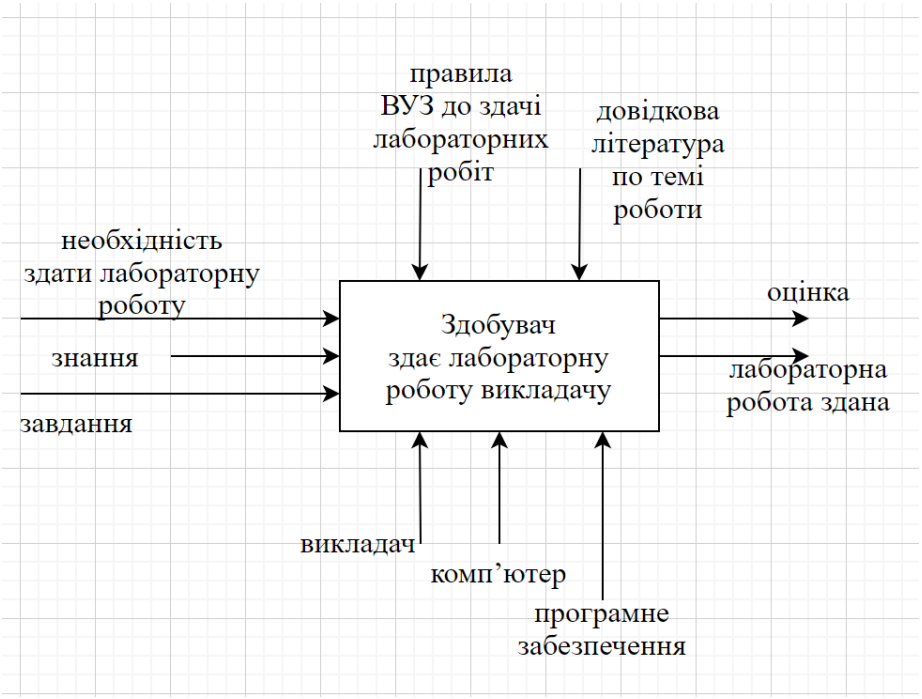


Рисунок 1.2 – Опис процесу «Здобувач здає лабораторну роботу викладачу» методом SADT

2. Завдання до лабораторних робіт 1

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 за допомогою текстового, табличного способу опису та згідно методології SADT (тільки діаграма верхнього рівня).

Лабораторні роботи 2. 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.

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.

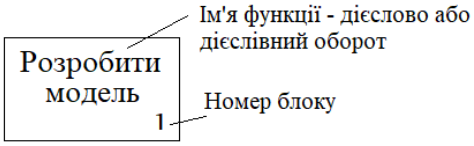
Стосовно до вже існуючих систем IDEF0 може бути використана для аналізу функцій, виконуваних системою, і відображення механізмів, за допомогою яких ці функції виконуються.

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.



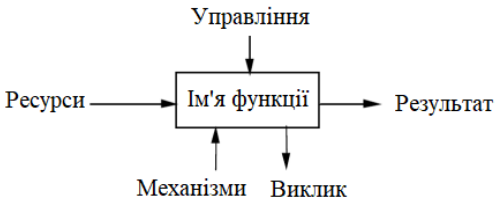
Стрілки являють дані або матеріальні об'єкти, пов'язані з функціями. Стрілки не уявляють потік або послідовність подій. Вони показують, які дані або матеріальні об'єкти мають надійти на вхід функції для того, щоб ця функція могла виконуватися.

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 2.1.

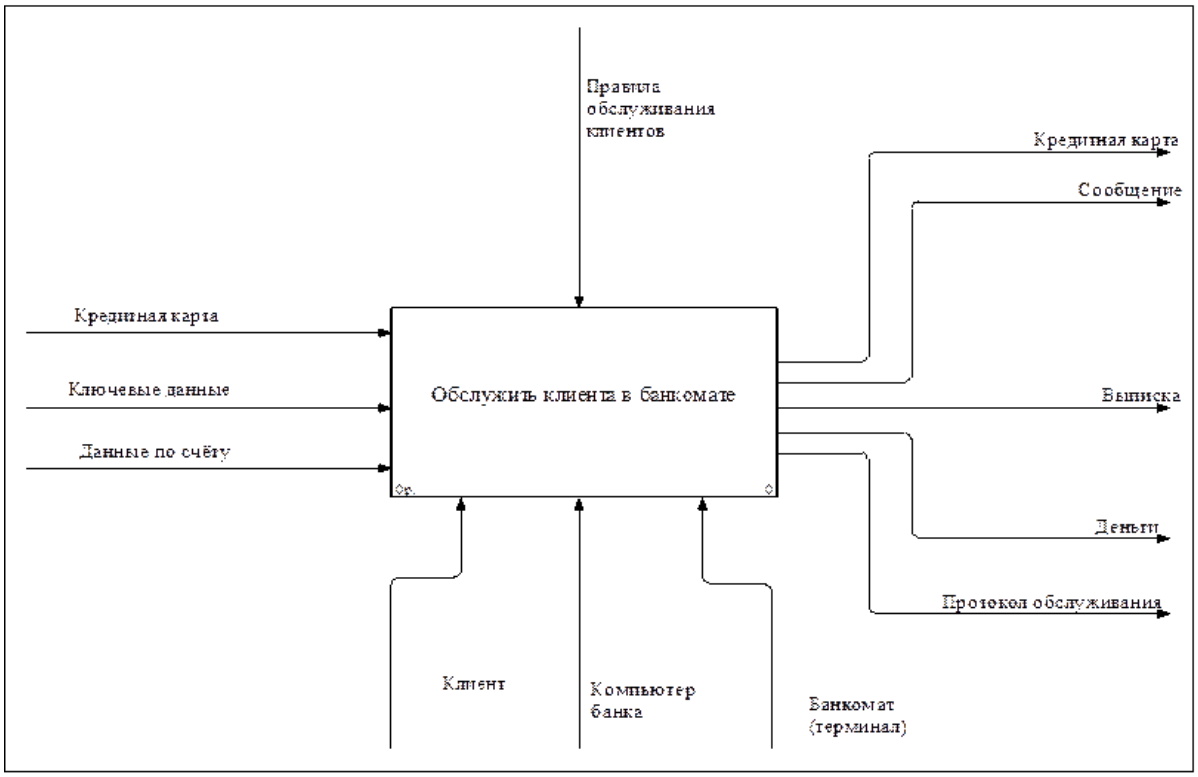


Рисунок 2.1 – Приклад процесу «Обслужити клієнта в банкоматі»

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 2.2).

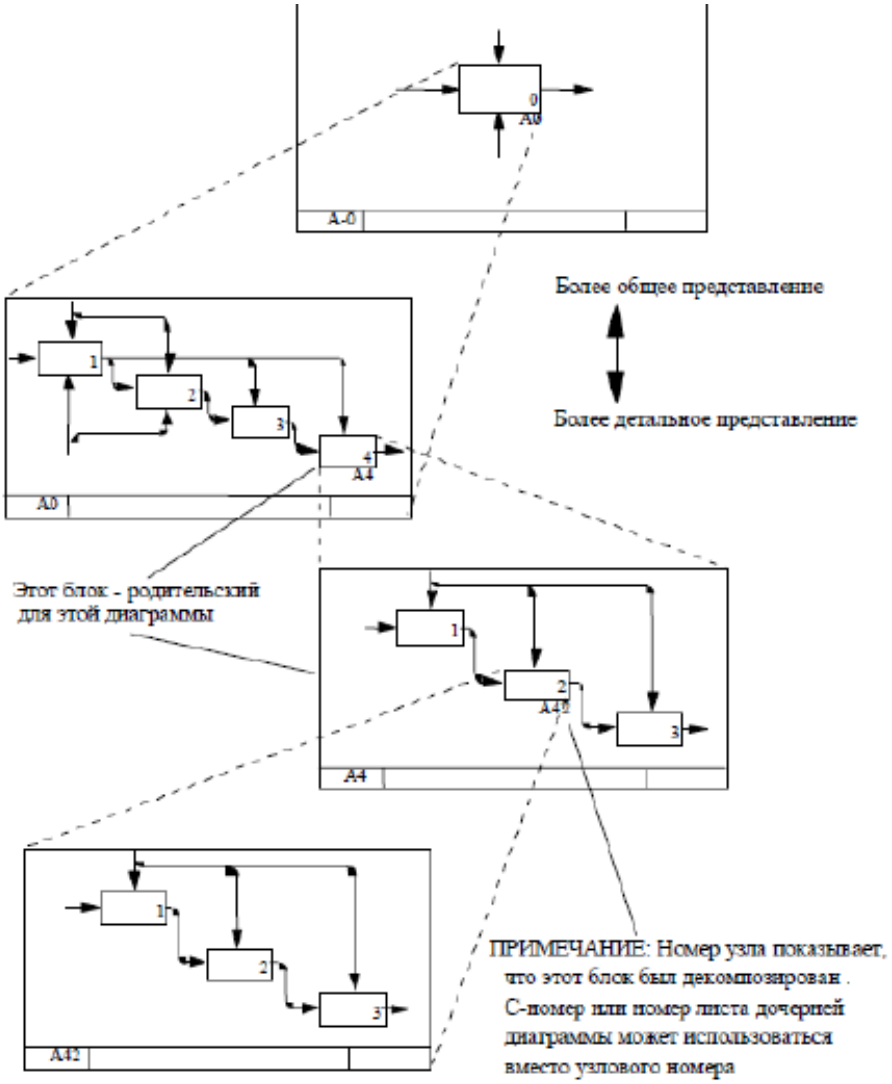
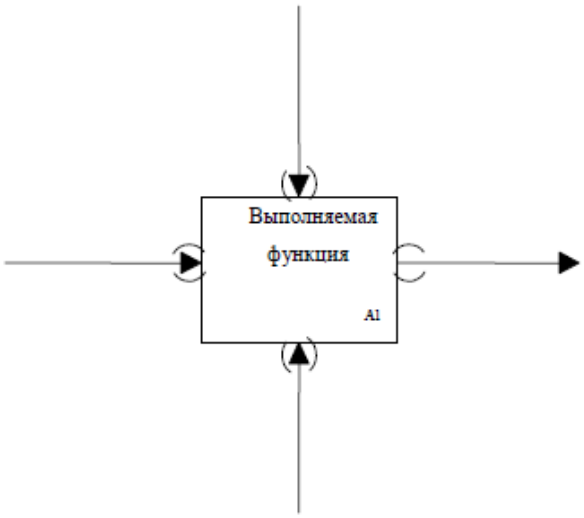


Рисунок 2.2 – Приклад декомпозиції

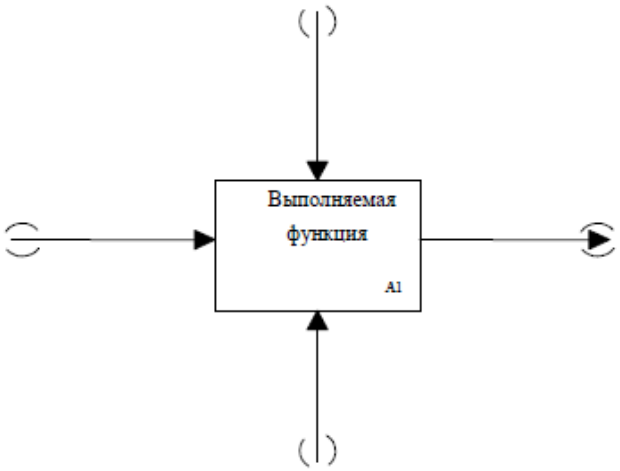
Ця діаграма, звана *діаграмою-нащадком*, описує все, пов'язане з цим блоком і його дугами, і не описує нічого поза цією межею. Декомпозіруємий блок називається *батьківським блоком*, а що містить його діаграма – *батьківської діаграмою*.

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 2.3 та 2.4.

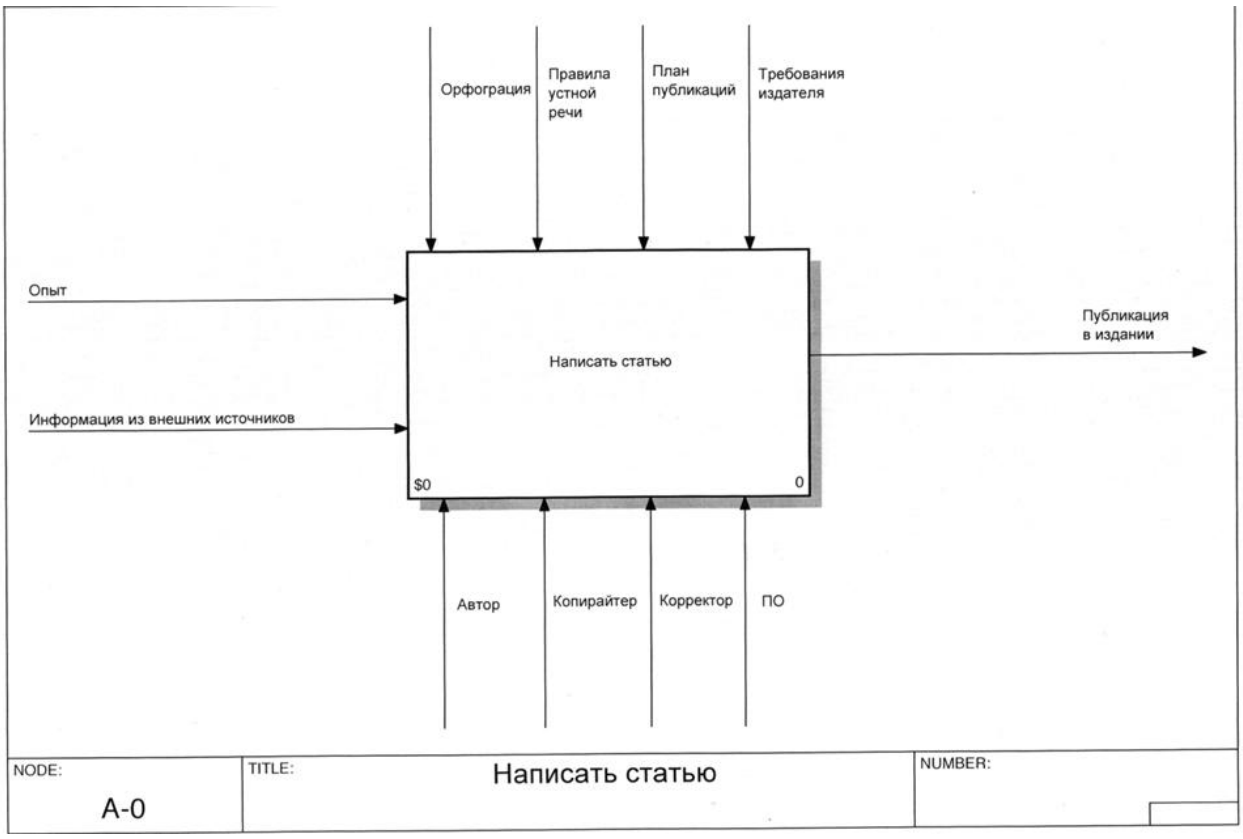


Рисунок 2.3 – Приклад контекстної діаграми А-0 процесу «Написання статті»

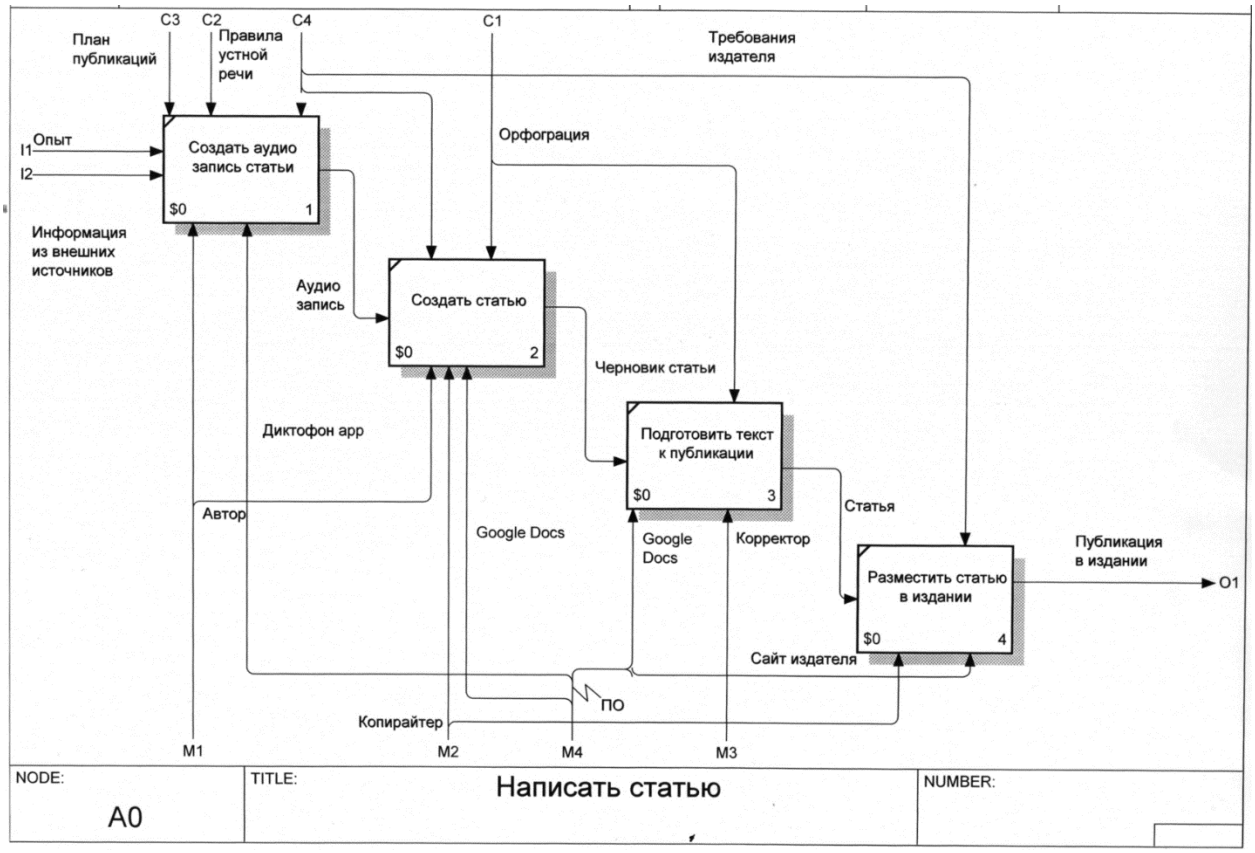


Рисунок 2.4 – Приклад контекстної діаграми А0 процесу «Написання статті»

2. Завдання до лабораторних робіт 2

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).

Лабораторна робота 3. 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.

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 використовуються 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 межах 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 3.1):

- 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 віддієслівними іменниками. Кожній дії призначається унікальний номер. Номер ідентифікатора процесу призначається послідовно.

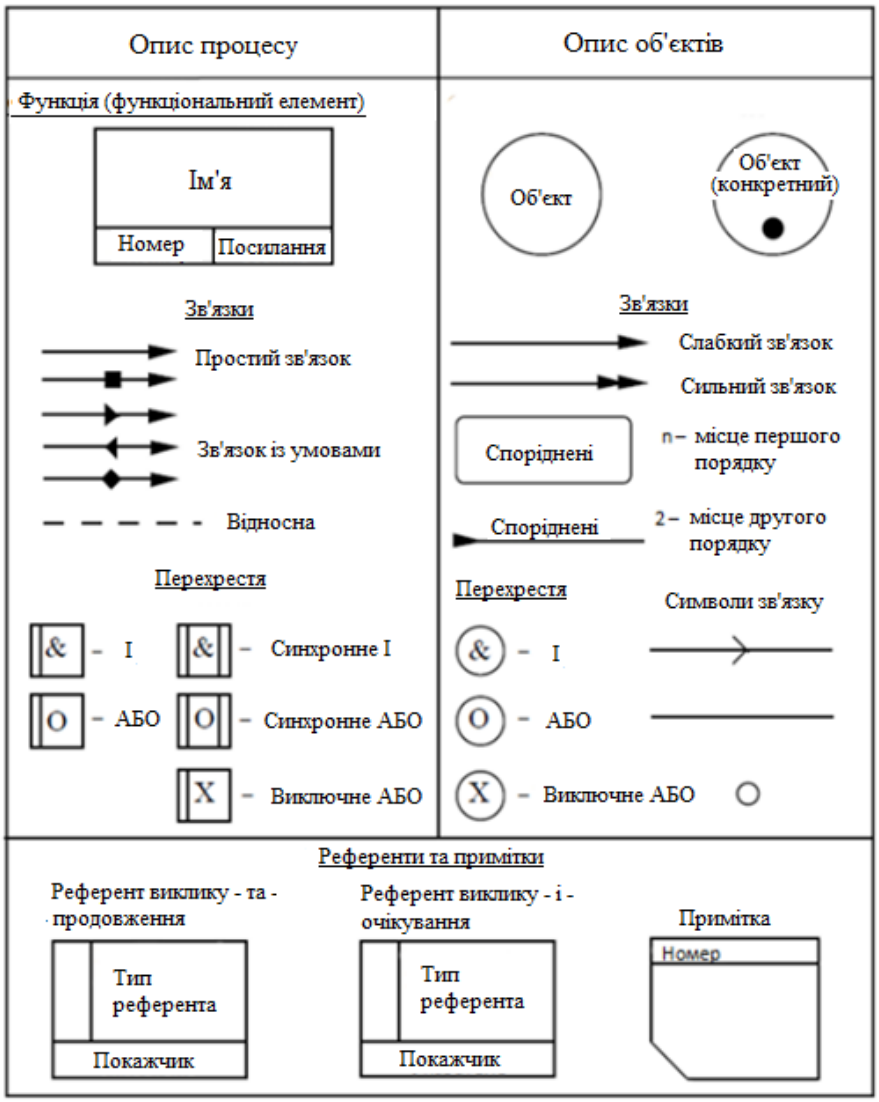
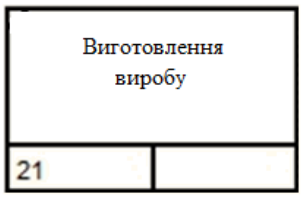


Рисунок 3.1 – Позначення основних елементів IDEF3-моделі

Приклад функції:



У правому нижньому кутку UOB-елемента розташовується посилання, яке використовується для вказівки посилань або на елементи з функціональної моделі IDEF0, або 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 часової 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.

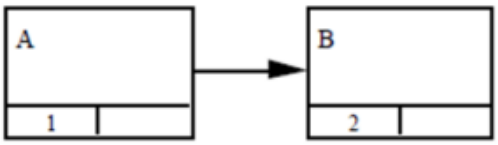
Існує два основних типи зв'язків, які використовуються в IDEF3 схемах:

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

Зв'язки простої черговості демонструють тимчасовий 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:

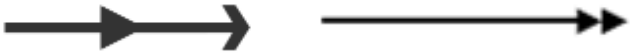


Приклад застосування:



Існують наступні підвиди зв'язку простої черговості:

Зв'язок *Об'єктний потік* застосовується в разі, коли об'єкт, що є результатом виконання вихідної дії, необхідний для виконання кінцевої 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 за послідовністю) може бути виконаний раніше елемента А (першого за порядком). Більше 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 неможливе та безглузде, 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 (стрілки відносин). Цей тип посилань підкреслює існування (можливо, що обмежують) відносин 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.

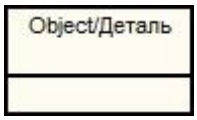
Для маркування використовуються абревіатури від назв типів зв'язків :

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

– СО або DL (від dashed links) – зв'язок 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 аспектів моделі. При внесенні об'єктів посилання, крім імені, слід вказувати тип об'єкта посилання.

Приклад позначення:



Приклад використання:

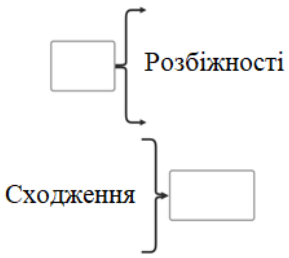


Для реалізації варіантів дій залежно від виконання певних умов були створені вузли.

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, залежно 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.



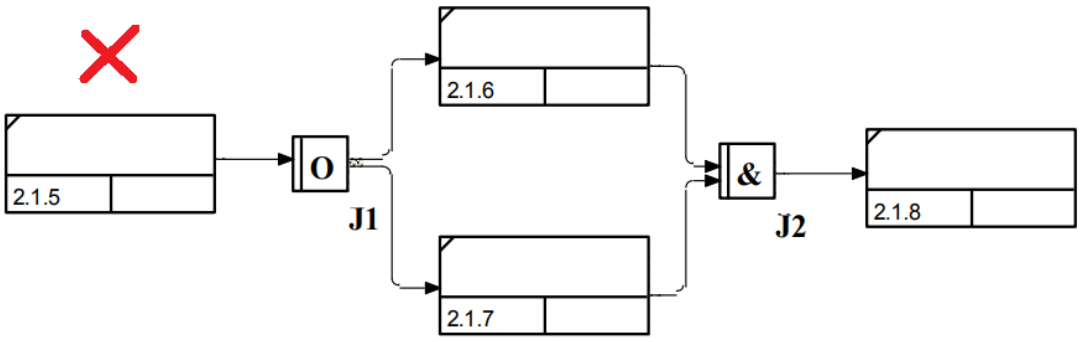
Синхронні вузли можуть бути не тільки вихідними, але й вхідними. В цьому випадку вузол активується тільки тоді, коли всі паралельні процеси закінчаться. Більше того, дуже важливо, щоб вони закінчилися одночасно.

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.

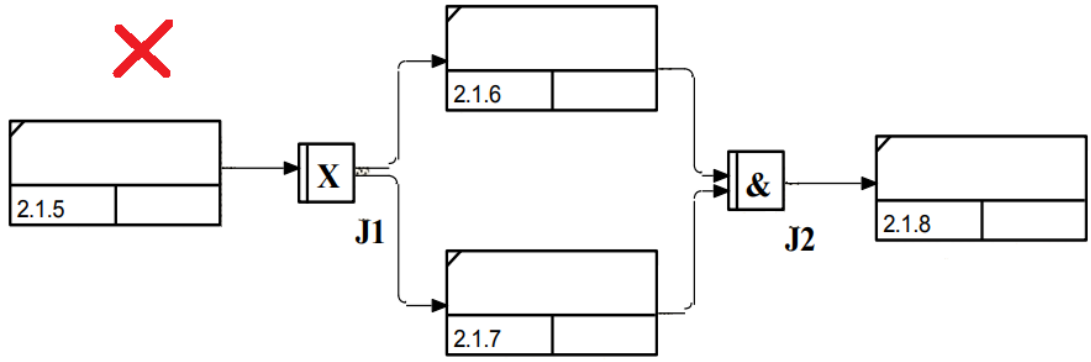
Всі з'єднання на діаграмі повинні бути парними. Однак при цьому типи з'єднань не зобов'язані збігатися. На діаграмі з'єднання зазвичай позначаються літерою «J» та цифрою.

Правила створення перехресть :

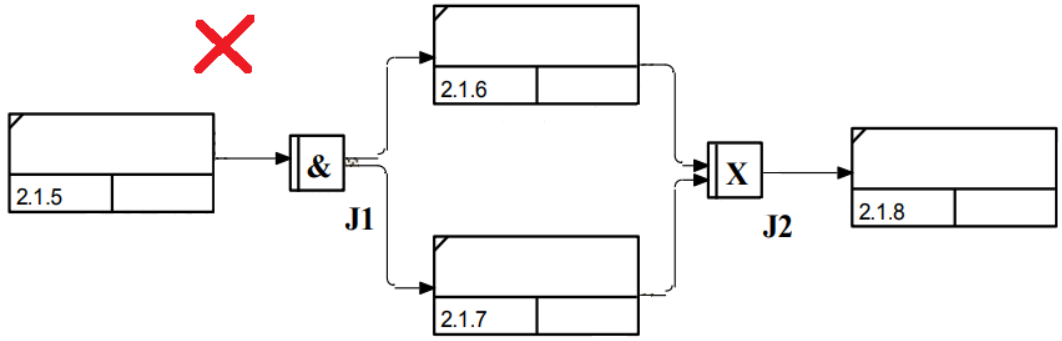
– Кожному перехрестю для злиття повинен передувати перехрестя для розгалуження.
– Перехрестя для злиття «І» не може слідувати за перехрестям для розгалуження типу синхронного або асинхронного «АБО».



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



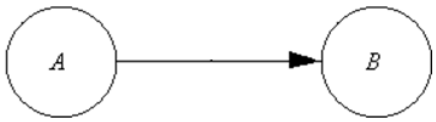
– Перехрестя для злиття типу виняткового «АБО» не може слідувати за перехрестям для розгалуження типу «І»



– Перехрестя, що має одну стрілку на одній стороні, повинно мати більше однієї стрілки на іншій.

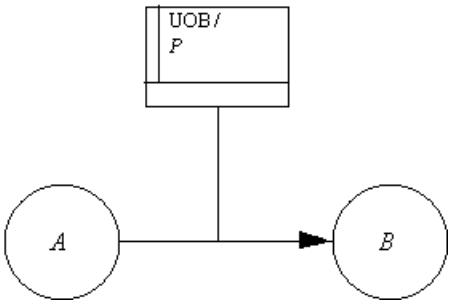
Якщо діаграми PFDD – технологічний процес «з точки зору спостерігача», то інший клас діаграм IDEF3 – *діаграми переходу стану об'єкта* (OSTN) – дозволяє розглядати той самий процес «з точки зору об'єкта».

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



Референти (Referents) 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.

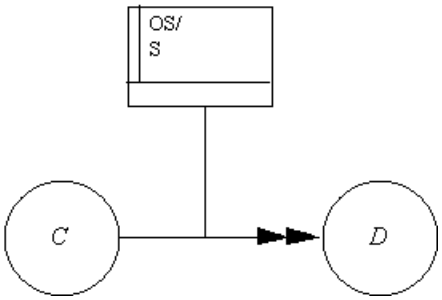
Наприклад:



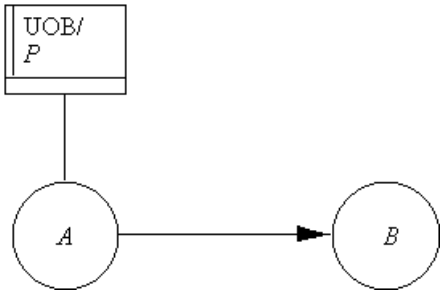
Існують декілька типів таких об'єктів:

– 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).

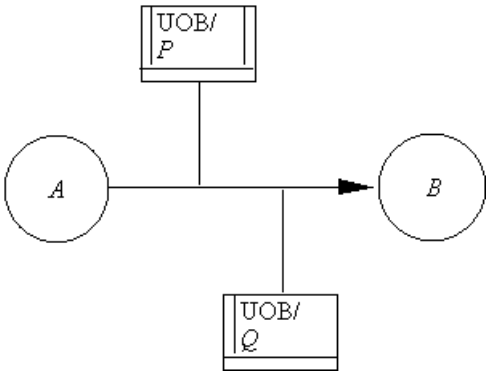
Приклад позначення «Викликай та продовжуй»



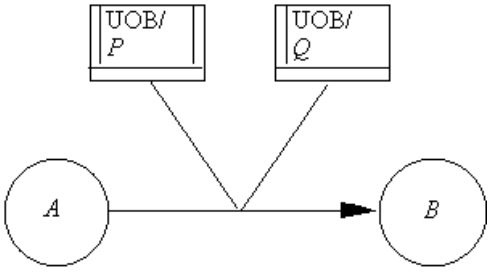
Існують референти, прикріплені до об'єктних станів. У даній ситуації досить часто використовується «утримання» об'єкта в даному стані; наприклад, в процесі заморожування дана речовина може підтримуватися в твердому стані. Ситуації цього типу можна представити конструкцією:



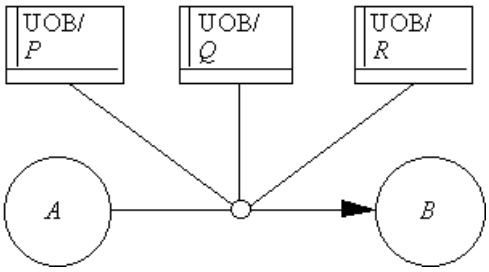
Схематики об'єктів з безліччю референтів. При переході з одного стану в інший часто може спостерігатися більш складний хід подій, в порівнянні з тим ходом подій, який може бути визначений одним референтом. Можна припускати, що деталі такого перебігу подій можуть бути забезпечені окремою схематикою процесів. Для цієї мети до однієї дуги 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 відсутнє (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 3.2) для завдання «Забарвлення деталі» та діаграму OSTN (рисунок 3.3) для цього ж завдання.

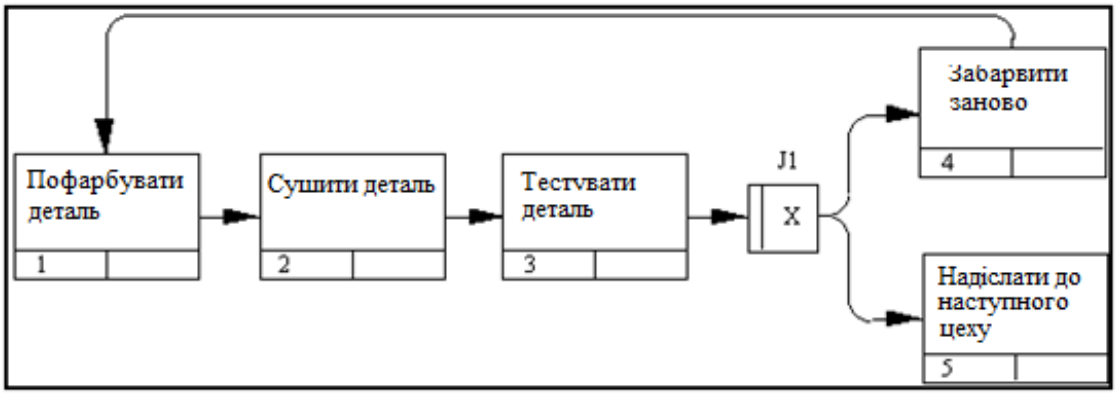


Рисунок 3.2 – Приклад діаграми PFDD

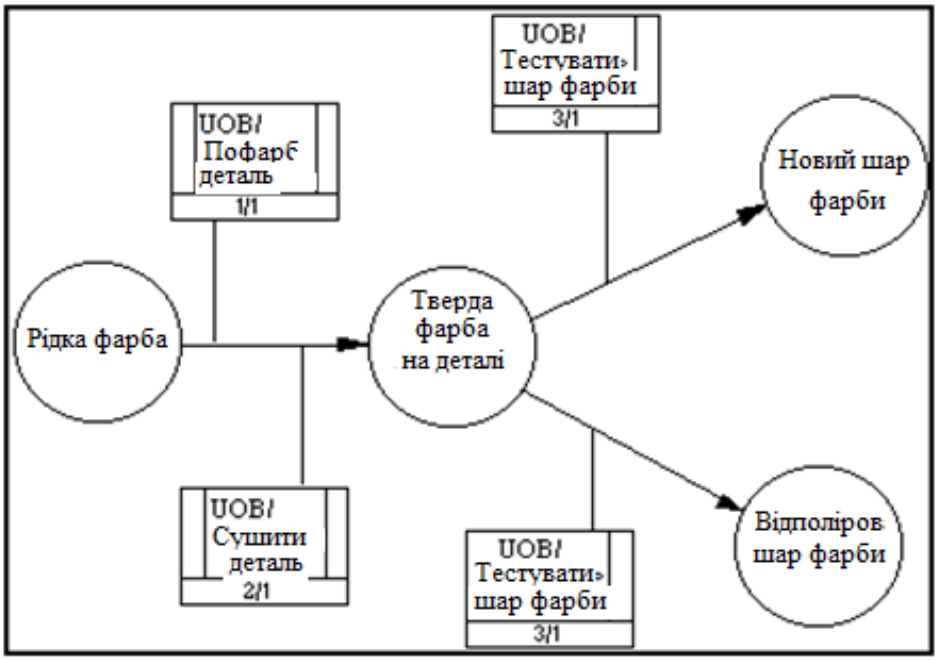


Рисунок 3.3 – Приклад діаграми OSTN

2. Завдання до лабораторної роботи 3

Оберіть задачу з області кібербезпеки, захисту інформації або іншої діяльності. Сформулювати обрану задачу. Обрати деякий процес із діаграм IDEF0 (з попередньої лабораторної роботи) та створити для обраного процесу діаграму опису послідовності етапів процесу (PFDD) та діаграму переходу стану об'єкта (OSTN) методології IDEF3.

Лабораторна робота 4. 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.

1. Теоретичний матеріал

IDEF1X є методом для розробки реляційних баз даних і використовує умовний синтаксис, спеціально розроблений для зручної 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 присвоюються унікальне ім'я і номер, розділяються косою рискою 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 сутність. 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 й того ж імені повинна завжди застосовуватися одна й та ж інтерпретація. Одна і та ж інтерпретація не може застосовуватися до різних імен, 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 як первинний ключ сутності або його частина, то сутність є залежною від ідентифікатора.
- Якщо використовується тільки частина зовнішнього ключа або взагалі не використовуються зовнішні ключі, то сутність є незалежною від ідентифікатора.

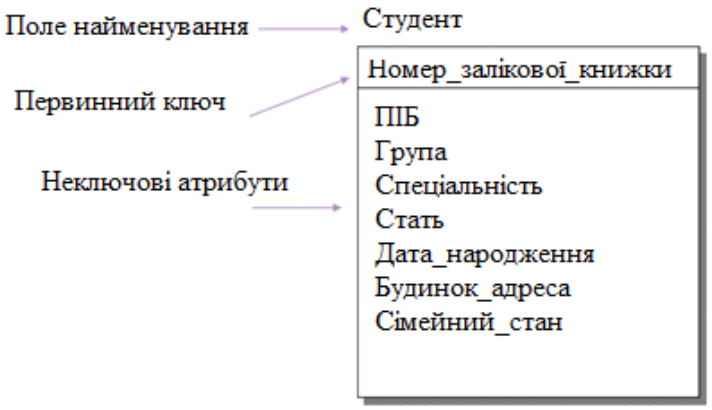
Атрибут представляє тип характеристик або властивостей, асоційованих з безліччю реальних або абстрактних об'єктів (людей, об'єктів, місць, подій, станів, ідей, пар предметів тощо).

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

Сутність повинна мати атрибут або комбінацію 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.

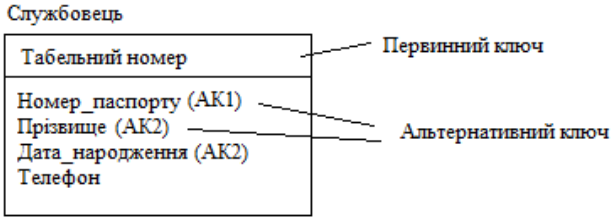
Успадковані передаються від сутності – батька при визначенні зв'язку.

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 як первинний ключ, а решта – як альтернативні ключі.

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

Наприклад:



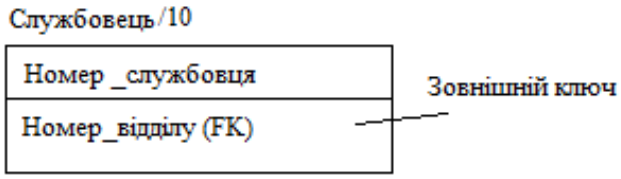
Якщо між двома сутностями є специфічне ставлення зв'язку або категоризації, то атрибути, що входять до первинного ключа батьківської або загальної суті, успадковуються як атрибути сутністю-нащадком. Ці успадковані атрибути називаються *зовнішніми ключами*.

Наслідуваний атрибут може використовуватися по суті як частина або ціле первинного ключа, альтернативного ключа або неключового 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 неідентифікованим.

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 проєктом пов'язано одне або більше завдань і завдання однозначно ідентифікуються тільки в межах свого проєкту, 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.

Специфічне 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 до точки.

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

Якщо потужність в точності дорівнює деякому числу N, це число (ціле, позитивне) поміщається близько до 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 «Проєкт складається з одного або більше завдань» може бути виведено з відносини, що зображує ПРОЕКТ як сутність – батько, ЗАВДАННЯ – як сутність – нащадок, з символом потужності P, 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 загальною 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*.

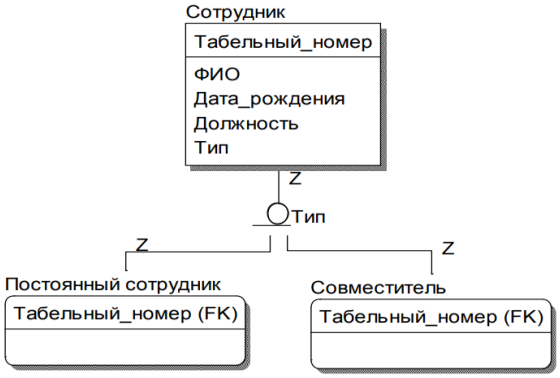
Відношення категоризації зображується лінією, що веде з загальної суті до підкресленого кола. Окремі лінії ведуть від підкресленого кола до кожної з сутностей-категорій.

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

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

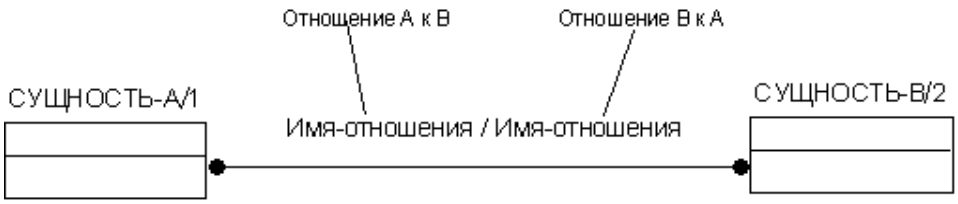
Ім'я атрибута загальної суті використовується як дискримінатор і записується поряд з колом.

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 4.1):

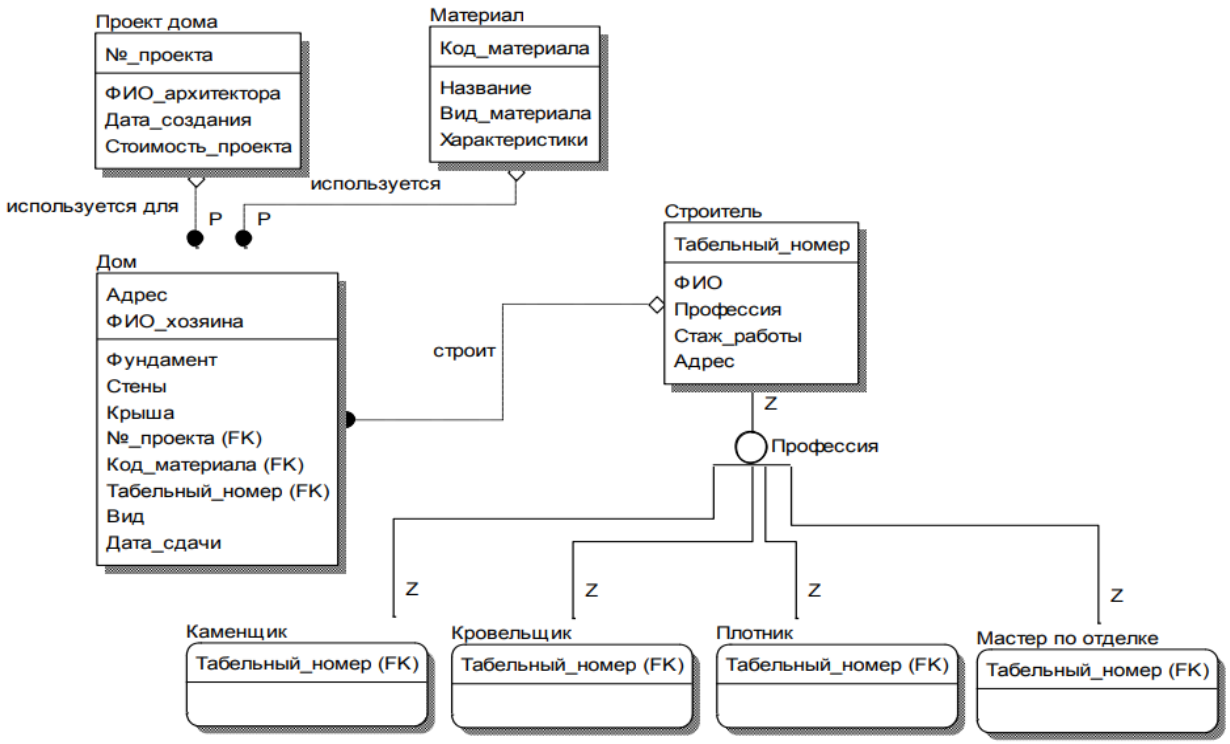


Рисунок 4.1 – Інформаційна модель процесу спорудження будиночка

2. Завдання до лабораторної роботи 4

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 з методологією IDEF1x. Оформити описану задачу згідно з методологією ERD. Порівняти отримані результати.

Лабораторна робота 5. Дослідження структури та умов використання методології UML та діаграми класів

Мета роботи:

Розглянути визначення бізнес-процесу та спосіб його опису за допомогою методології UML. Навчитися розробляти задачі у вигляді діаграми класів, описаної за методологією UML.

1. Теоретичний матеріал

Діаграма класів (class diagram) – статична структурна діаграма, що описує структуру системи. Вона демонструє класи системи, їх атрибути, методи і залежності між класами. Це один з найбільш часто використовуваних видів діаграм UML. Зазвичай створення діаграми класів знаменує собою завершення процесу аналізу та початок процесу проєктування.

Діаграма класів складається з безлічі елементів, які в сукупності відображають декларативні знання про предметну область. Ці знання інтерпретуються в базових поняттях мови UML, таких як класи, інтерфейси і відносини між ними і їх складовими компонентами. При цьому окремі компоненти цієї діаграми можуть утворювати пакети для представлення більш загальної моделі системи. Якщо діаграма класів є частиною деякого пакета, то її компоненти повинні відповідати елементам цього пакета, включаючи можливі посилання на елементи з інших пакетів.

Клас (class) в мові UML служить для позначення безлічі об'єктів, які володіють однаковою структурою, поведінкою і відносинами з об'єктами з інших класів. Графічно клас зображується у вигляді прямокутника, який додатково може бути розділений горизонтальними лініями на розділи або секції (рисунок 4.1). У цих розділах можуть зазначатися ім'я класу, атрибути (змінні) і операції (методи).



а б в

Рисунок 5.1 – Графічне зображення класу на діаграмі класів: а – тільки ім'я класу; б – ім'я класу і його атрибути; в – ім'я класу, його атрибути та операції

Інтерфейс відображає зовнішні прояви об'єкта, показує, яким чином здійснюється взаємодія з ним, приховуючи інші деталі, що не мають відношення до процесу взаємодії. Об'єкт є окремим екземпляром класу, який створюється на етапі виконання програми. Він має своє власне ім'я і конкретні значення атрибутів. Для зображення інтерфейсу використовується спеціальний графічний символ – прямокутник класу з ключовим словом або стереотипом "interface". При цьому секція атрибутів у прямокутнику відсутня, а вказується лише секція операцій (рисунок 5.2).

Інтерфейс завжди реалізується деяким класом, який в такому випадку називають класом, що підтримує інтерфейс. Один і той же об'єкт може мати кілька інтерфейсів. Це означає, що клас цього об'єкта реалізує всі операції цих інтерфейсів.



Рисунок 5.2 – Графічне зображення інтерфейсу на діаграмі класів

Шаблон (template) або *параметризований клас (parametrized class)* призначений для позначення такого класу, який має один (або більше) нефіксований формальний параметр. Він

визначає ціле сімейство або безліч класів, кожен з яких може бути отриманий зв'язуванням цих параметрів з дійсними значеннями. Зазвичай параметрами шаблонів служать типи атрибутів класів, такі як цілі числа, перерахування, масив рядків і інші. В більш складному випадку формальні параметри можуть представляти і операції класу.

Графічно шаблон зображується прямокутником, до верхнього правого кута якого приєднаний маленький прямокутник з пунктирних ліній. Великий прямокутник може бути розділений на секції, аналогічно позначенню для класу (рисунок 5.3). У верхньому прямокутнику вказується список формальних параметрів для тих класів, які можуть бути отримані на основі даного шаблону. У верхній секції шаблону записується його ім'я за правилами запису імен для класів.

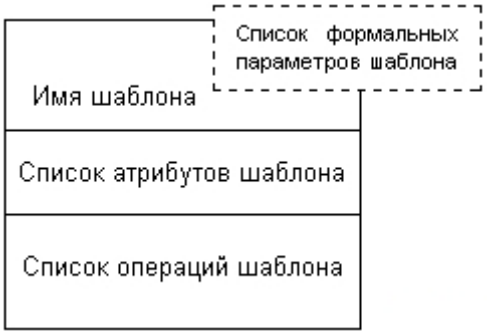


Рисунок 5.3 – Графічне зображення шаблону на діаграмі класів

Об'єкт (object) є окремим екземпляром класу, який створюється на етапі виконання програми. Він має своє власне ім'я і конкретні значення атрибутів. В силу різних причин може виникнути необхідність показати взаємозв'язки не тільки між класами моделі, а й між окремими об'єктами, що реалізують ці класи. Для графічного зображення об'єктів використовується такий же символ прямокутника, що і для класів. Відмінності проявляються при вказівці імен об'єктів, які в разі об'єктів обов'язково підкреслюються (рисунок 4.4).

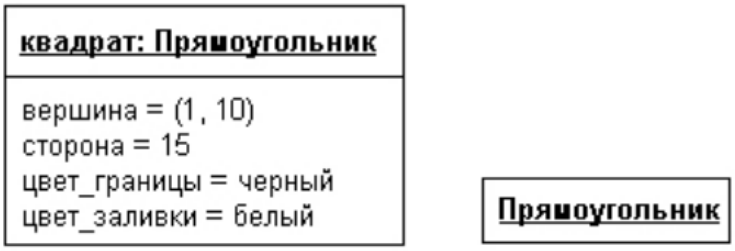


Рисунок 5.4 – Приклади об'єктів класу

Розглянемо докладніше побудову, структуру і позначення класів.

Ім'я класу має бути унікальним в межах пакета, який описується деякою сукупністю діаграм класів (можливо, однією діаграмою). Воно вказується в першій верхній секції прямокутника (див. рисунок 4.1а). На додаток до загального правила найменування елементів мови UML, ім'я класу записується по центру секції імені напівжирним шрифтом і має починатися з великої літери.

Клас може не мати екземплярів або об'єктів. У цьому випадку він називається абстрактним класом, а для позначення його імені використовується похилий шрифт (курсив). У мові UML прийнято спільну угоду про те, що будь-який текст, що відноситься до абстрактного елемента, записується курсивом. Дана обставина є семантичним аспектом опису відповідних елементів мови UML.

У другій зверху секції прямокутника класу записуються його атрибути (attributes) або властивості. У мові UML прийнята певна стандартизація запису атрибутів класу, яка підпорядковується деяким синтаксичним правилам. Кожному атрибуту класу відповідає окремий рядок тексту, який складається з квантора видимості атрибута, імені атрибута, його кратності, типу значень атрибута і, можливо, його початкового значення.

Квантор видимості може приймати одне з трьох можливих значень і, відповідно, відображається за допомогою спеціальних символів (рисунок 5.5):

– Символ "+" позначає атрибут з областю видимості типу загальнодоступний (public).

Атрибут із цією областю видимості доступний або видний з будь-якого іншого класу пакета, в якому визначена діаграма.

– Символ "#" позначає атрибут з областю видимості типу захищений (protected). Атрибут із цією областю видимості недоступний або непомітний для всіх класів, за винятком підкласів даного класу.
– Символ "-" позначає атрибут з областю видимості типу закритий (private). Атрибут із цією областю видимості недоступний або непомітний для всіх класів без винятку.
– Символ "~" – позначає атрибут з областю видимості типу пакетний (package). Атрибут із цією областю видимості недоступний або непомітний для всіх класів за межами пакета, в якому визначено клас-власник даного атрибута.

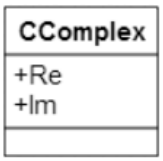


Рисунок 5.5 – Приклад класу з кванторами видимості в атрибутах

Операція (operation) являє собою деякий сервіс, що надає кожний екземпляр класу за певною вимогою. Сукупність операцій характеризує функціональний аспект поведінки класу. При цьому кожній операції класу відповідає окремий рядок, який складається з квантора видимості операції, імені операції, типу значення операції і, можливо, рядок-властивість даної операції.

Для підвищення продуктивності системи одні операції можуть виконуватися паралельно або одночасно, а інші – тільки послідовно. В цьому випадку для вказівки паралельності виконання операції використовується рядок-властивість виду «{concurrency = ім'я}», де ім'я може приймати одне з наступних значень:

– *послідовна (sequential)* – для даної операції необхідно забезпечити її єдине виконання в системі, одночасне виконання інших операцій може призвести до помилок або порушень цілісності об'єктів класу.
– *паралельна (concurrent)* – дана операція в силу своїх особливостей може виконуватися паралельно з іншими операціями в системі, при цьому паралельність повинна підтримуватися на рівні реалізації моделі.
– *охоронюване (guarded)* – всі звернення до даної операції повинні бути строго впорядковані в часі з метою збереження цілісності об'єктів даного класу, при цьому можуть бути прийняті додаткові заходи по контролю виняткових ситуацій на етапі її виконання.

Приклад класу з операціями продемонстрований на рисунку 5.6.

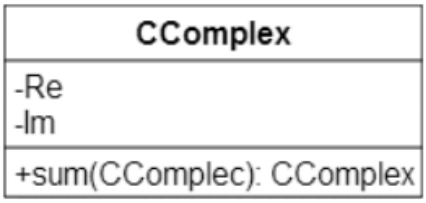


Рисунок 5.6 – Приклад класу з кванторами видимості в операціях

Крім внутрішнього устрою або структури класів на відповідній діаграмі вказуються різні відносини між класами. При цьому сукупність типів таких відносин фіксована в мові UML і зумовлена семантикою цих типів відносин. Базовими відносинами або зв'язками в мові UML є (рисунок 5.7):

– Відношення залежності (dependency relationship)
– Відношення асоціації (association relationship)
– Відношення узагальнення (generalization relationship).

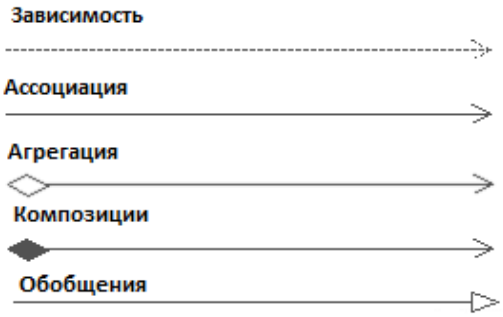


Рисунок 5.7 – Графічне зображення відносин на діаграмі класів

Відношення залежності в загальному випадку вказує на деяке семантичне відношення між двома елементами моделі або двома множинами таких елементів. Ставлення залежності використовується в такій ситуації, коли деяка зміна одного елемента моделі може вимагати зміни іншого залежного від нього елемента моделі (рисунок 5.8).

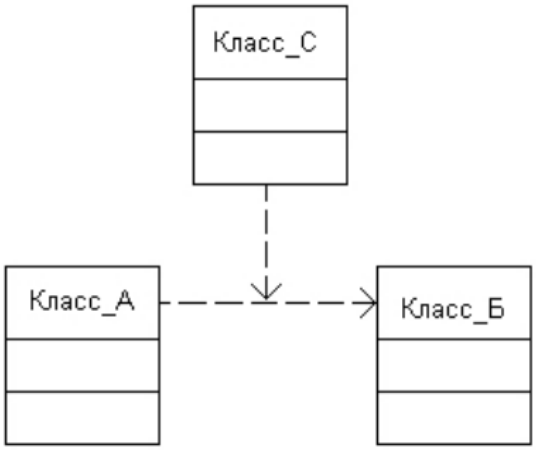


Рисунок 5.8 – Приклад позначення відносини залежності

Відношення асоціації відповідає наявності деякого відношення між класами. Дане відношення позначається суцільною лінією з додатковими спеціальними символами, які характеризують окремі властивості конкретної асоціації (рисунок 5.9).



Рисунок 5.9 – Приклад позначення відносини асоціації

Відношення узагальнення є звичайним таксономічним ставленням між більш загальним елементом (батьком або предком) і більш приватним або спеціальним елементом (дочірнім або нащадком). Дане відношення може використовуватися для подання взаємозв'язків між пакетами, класами, варіантами використання і іншими елементами мови UML (рисунок 5.10).

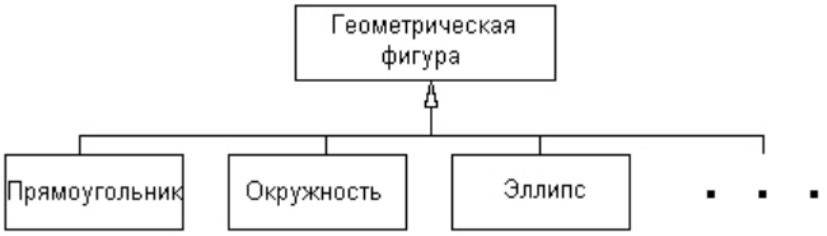


Рисунок 5.10 – Приклад позначення відносини узагальнення

Стосовно діаграми класів, дане відношення описує ієрархічну будову класів та успадкування їх властивостей і поведінки. При цьому передбачається, що клас-нащадок має всі властивості і поведінку класу-предка, а також має свої власні властивості і поведінку, які відсутні у класу-предка.

Поруч зі стрілкою узагальнення може розміщуватися рядок тексту, який вказує на деякі додаткові властивості цього відношення. Цей текст буде стосуватися всіх ліній узагальнення, які йдуть до класів-нащадків. Іншими словами, відзначена властивість стосується всіх підкласів даної відносини. При цьому текст слід розглядати як обмеження, і тоді він записується у фігурних дужках. В якості обмежень можуть бути використані наступні ключові слова мови UML:

– {complete} – означає, що в даному відношенні узагальнення специфіковані усі класи-нащадки, і інших класів-нащадків у даного класу-предка бути не може. Приклад – клас Кліент_банку є предком для двох класів: Фізічне_ліцо і Компанія, і інших класів-нащадків він не має. На відповідній діаграмі класів це можна вказати явно, записавши поруч з лінією узагальнення дану строку – обмеження.
– {disjoint} – означає, що класи-нащадки не можуть містити об'єктів, які одночасно є екземплярами двох або більше класів. У наведеному вище прикладі ця умова також виконується, оскільки передбачається, що жодна конкретна фізична особа не може бути одночасно і конкретною компанією. У цьому випадку поряд з лінією узагальнення можна записати цей рядок-обмеження;
– {incomplete} – означає випадок, протилежний першому. А саме, передбачається, що на діаграмі вказані в повному обсязі класи-нащадки. У подальшому можливе поповнити їх перелік, не змінюючи вже побудовану діаграму. Приклад діаграми класу «Автомобіль», для якої зазначення всіх без винятку моделей автомобілів можна порівняти зі створенням відповідного каталогу. З іншого боку, для окремого завдання, такого як розробка системи продажу автомобілів конкретних моделей, в цьому немає необхідності. Але вказати неповноту структури класів-нащадків все ж слід.
– {overlapping} – означає, що окремі екземпляри класів-нащадків можуть належати одночасно декільком класам. Приклад – клас «Багатокутник» є класом-предком для класу «Прямокутник» і класу «Ромб». Однак існує окремий клас «Квадрат», екземпляри якого одночасно є об'єктами перших двох класів. Цілком природно таку ситуацію вказати явно за допомогою цього рядка-обмеження.

Приклад використання рядка-обмеження щодо узагальнення продемонстровано на рисунку 5.11.

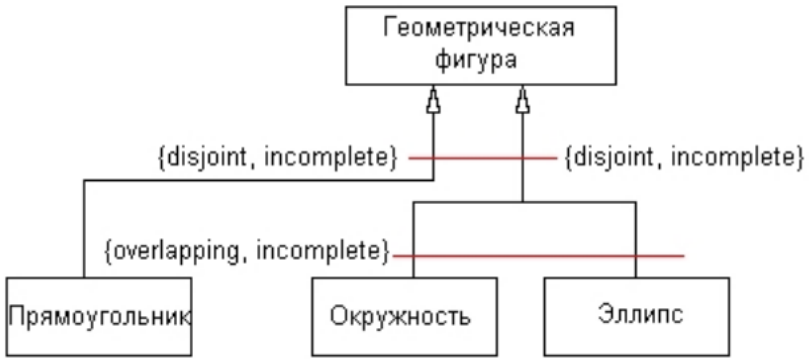


Рисунок 5.11 – Приклад позначення відносини узагальнення

Відношення агрегації має місце між декількома класами в тому випадку, якщо один з класів є

деякою сутністю, що включає в себе в якості складових частин інші сутності («частина-ціле»). Частини системи не повинні наслідувати її властивості і поведінку, оскільки є цілком самостійними сутностями. Більше того, частини цілого мають свої власні атрибути та операції, які істотно відрізняються від атрибутів і операцій цілого.

Відношення композиції є окремим випадком відношення агрегації. Це відношення служить для виділення спеціальної форми відносини "частина-ціле", при якій складові частини в деякому сенсі перебувають всередині цілого. Специфіка взаємозв'язку між ними полягає в тому, що частини не можуть виступати у відриві від цілого, тобто зі знищенням цілого знищуються і всі його складові частини.

На рисунку 5.12 показано приклад діаграми класів.

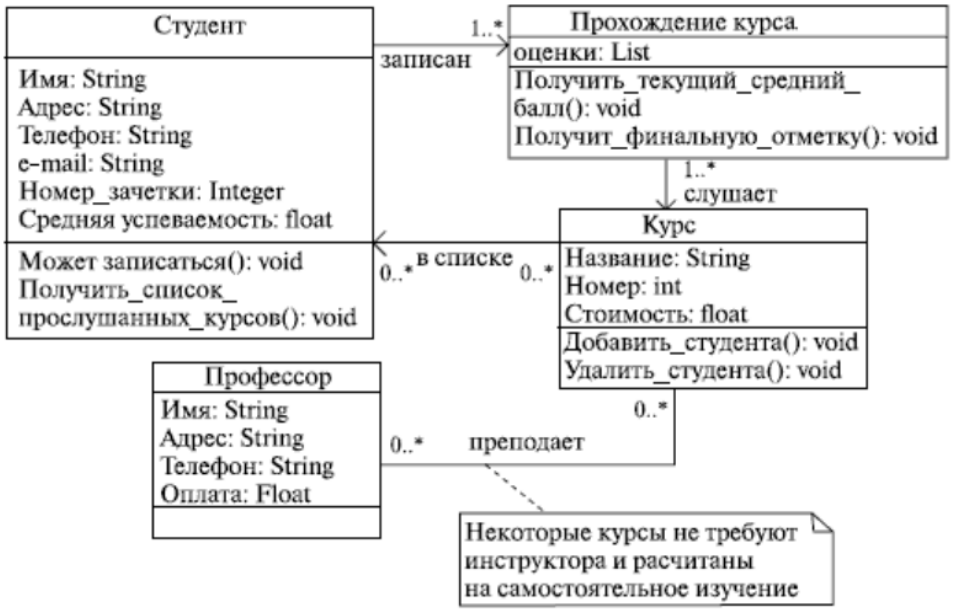


Рисунок 5.12 – Приклад діаграми класів

Розглянемо основні кроки, які необхідно виконати в програмі ArgoUML.

Запустимо ArgoUML. З'явиться вікно, зображене на рисунку 5.13.

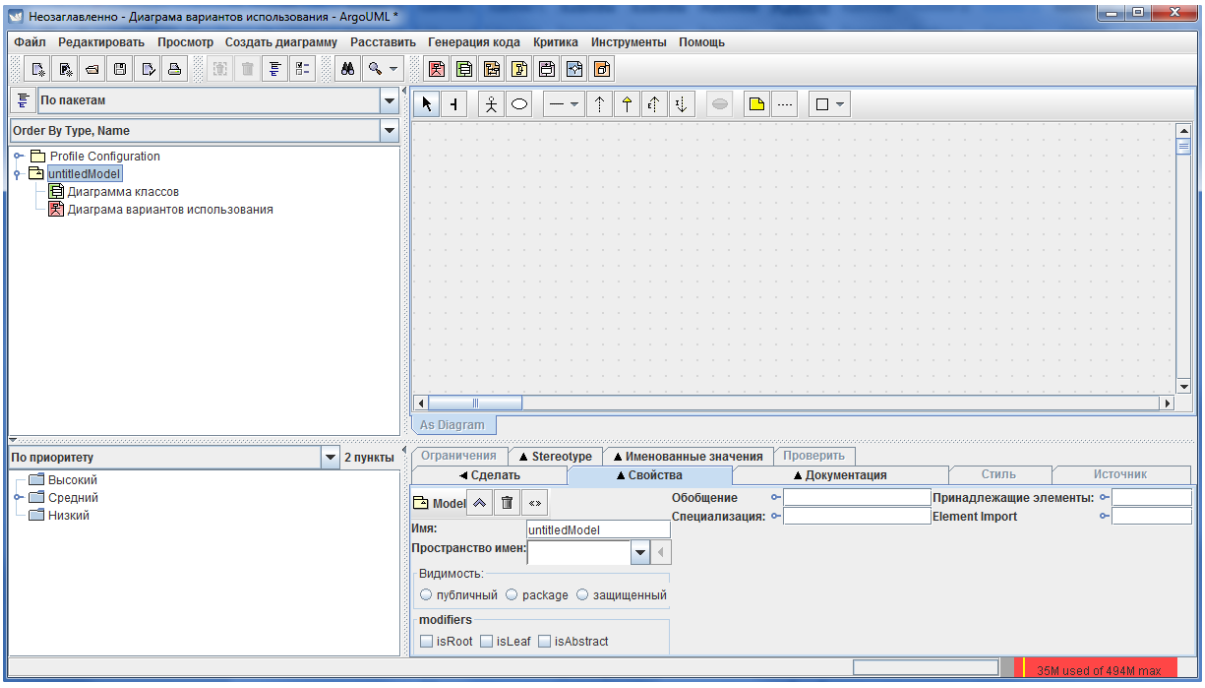


Рисунок 5.13 – Вікно програми ArgoUML

При створенні нового проєкту створюється модель під назвою untitledModel. Замість цього

імені можна ввести будь-яке інше, наприклад, MathTasks (рисунок 5.14).

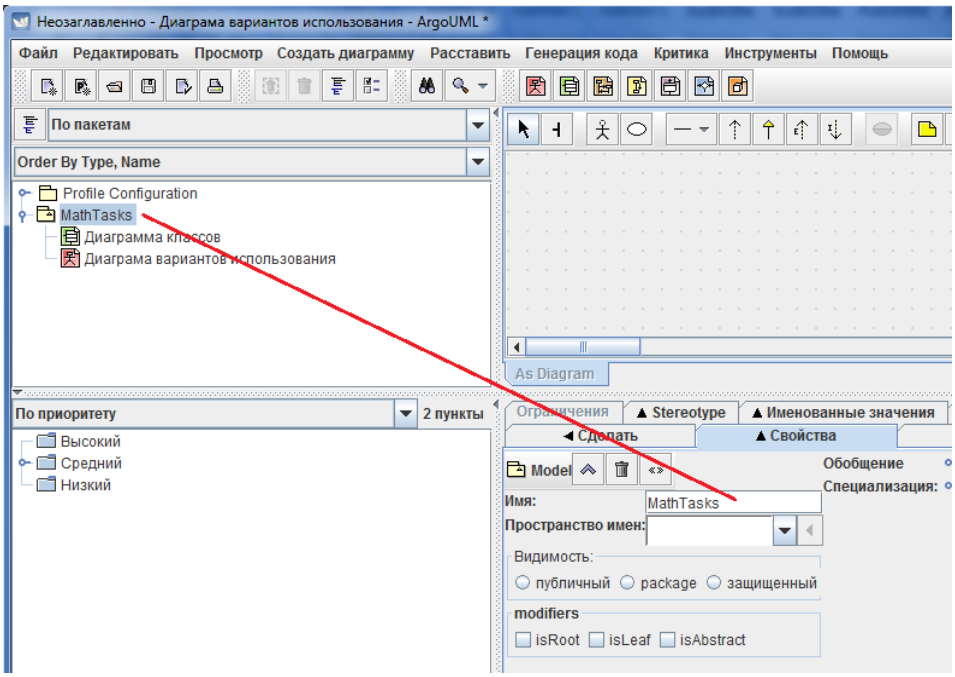


Рисунок 5.14 – Зміна імені моделі

Меню «Створити діаграму» містить перелік діаграм, які можна створити для цього проєкту. Також потрібну діаграму можна вибрати на панелі інструментів.



Нехай розробляємо систему обробки замовлень.

Створюємо головну діаграму класів. Для цього двічі клацніть на значку UntitledModel, відобразиться вміст пакета. Виберіть «Діаграма класів». За допомогою кнопки «Новий пакет» і панелі інструментів розмістіть на діаграмі новий пакет. У вкладці Властивості вкажіть ім'я пакета Entities (Сутність) (рисунок 5.15).

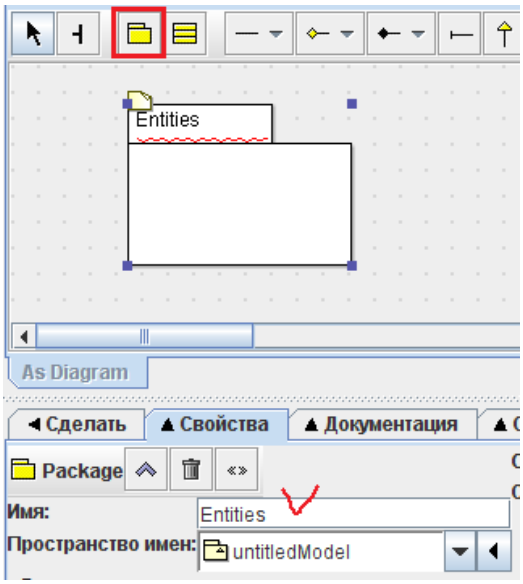
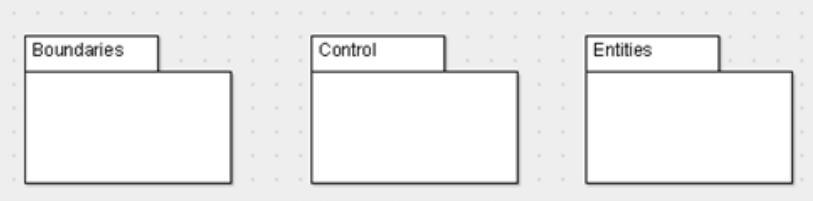
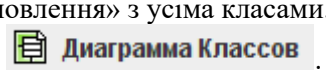


Рисунок 5.15 – Створення пакета в головній діаграмі класів

Аналогічним чином створюємо пакети Boundaries (Межі) і Control (Управління).



Створимо діаграми класів для сценарію «Ввести нове замовлення» з усіма класами.

У меню «Створити діаграму» виберіть «Діаграма класів» 

. У вкладці Властивості введіть ім'я для нової діаграми класу Add New Order (Введення нового замовлення). Клацніть в браузері на цій діаграмі, щоб відкрити її.

Створимо в вікні діаграми класів класи OrderOptions (Вибір замовлення). Для цього в панелі інструментів необхідно натиснути кнопку . Створимо нову операцію в створеному класі (рисунок 5.16).



Рисунок 5.16 – Створення нової операції

В поле Ім'я створюваної операції введемо Create. Для зазначення значення void створимо такий тип даних. Для цього клацнемо правою кнопкою миші на значку UntitledModel і вибираємо пункт «Новий тип Даних» (рисунок 5.17)

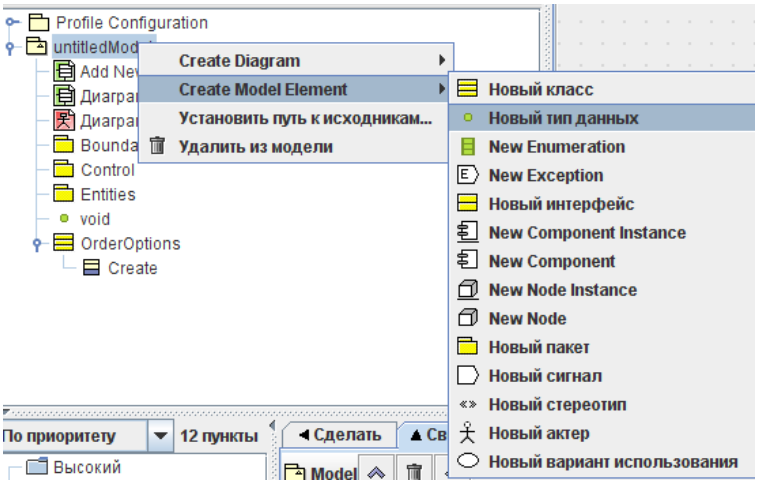


Рисунок 5.17 – Створення нового типу даних

У вкладці Властивості створюваної операції вибираємо Parameter і подвійним клацанням миші натискаємо на return. Потім в полі Тип вибираємо створене значення void (рисунок 5.18).

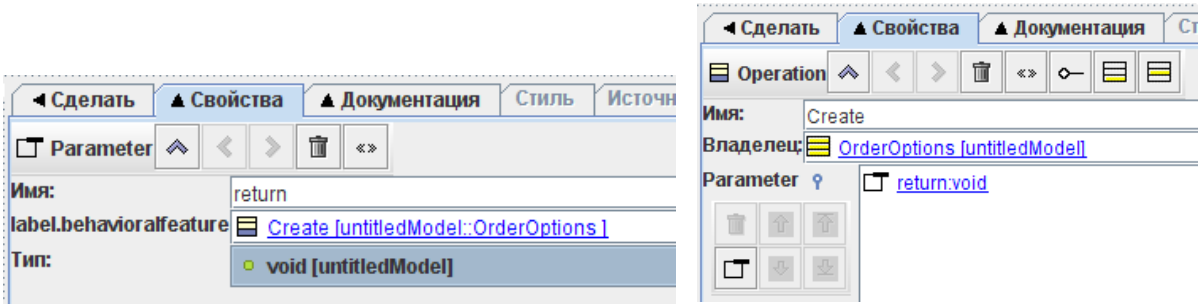
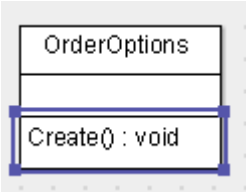


Рисунок 5.18 – Вибір типу значення, що повертається для операції

В результаті отримуємо наступний клас:



Аналогічним чином створюємо клас OrderDetail (Деталі замовлення) і його операції Open (): void, SubmitInfo (): void, Info (): void, клас Order (Замовлення) і його операції Create (): void, SetInfo (): void, GetInfo (): void, клас OrderMgr (Менеджер замовлень) і його операцію SaveOrder (): void і клас TransactionMgr (Менеджер транзакцій) і його операції SaveOrder():void, Commit():void.

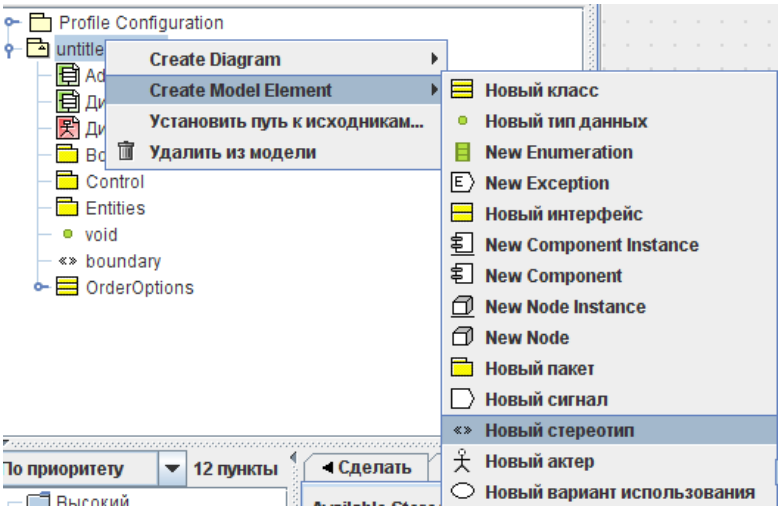
Імені класу може передувати стереотип.

Стереотипи є одним із трьох типів механізмів розширюваності в уніфікованій мові моделювання (UML). Вони дозволяють проєктувальникам розширювати словник UML для створення нових елементів моделювання, одержуваних з існуючих, але мають певні властивості, які підходять для конкретної проблеми предметної області або для іншого спеціалізованого використання. Графічно стереотип відображається як ім'я, укладене в лапки («», або, якщо такі лапки неприпустимі, << >>) і розташоване над ім'ям іншого елемента.

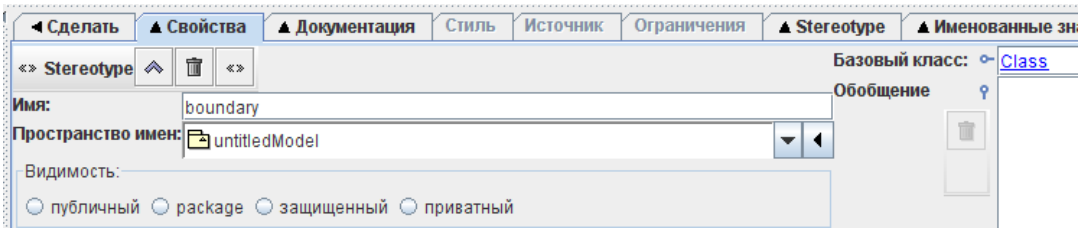
У цій таблиці наведено стандартні стереотипи класів.

Стереотип	Описание
«actor»	Действующее лицо
«auxiliary»	Вспомогательный класс
«enumeration»	Перечислимый тип данных
«exception»	Исключение (только в UML 1)
«focus»	Основной класс
«implementationClass»	Реализация класса
«interface»	Все составляющие абстрактные
«metaclass»	Экземпляры являются классами
«powertype»	Метакласс, экземплярами которого являются все наследники данного класса (только в UML 1)
«process»	Активный класс
«thread»	Активный класс (только в UML 1)
«signal»	Класс, экземплярами которого являются сигналы
«stereotype»	Новый элемент на основе существующего
«type»	Тип данных
«dataType»	Тип данных
«utility»	Нет экземпляров, служба

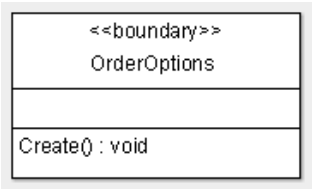
У нашому прикладі імена стереотипів будуть схожі на назви створених раніше пакетів. Додаймо стереотипи в класи: натискаємо на значку UntitledModel і вибираємо пункт «Новий стереотип».



В імені стереотипу пишемо boundary і вибираємо базовий клас Class.



Тепер натискаємо на клас OrderOptions і в закладці Stereotype вибираємо створений стереотип. В результаті отримаємо:



Доповніть створені класи стереотипами, як показано на рисунку 5.19.

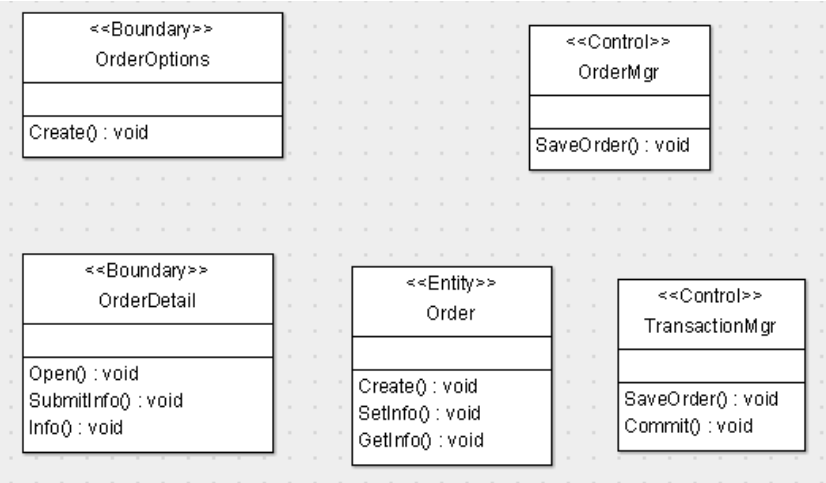


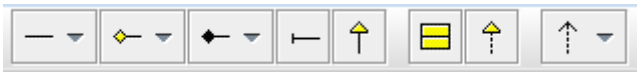
Рисунок 5.19 – Стереотипи та класи, що створюються

Об'єднаймо класи в пакети. Для цього перетягніть в браузері клас OrderDetail на пакет Boundaries, клас OrderOptions на пакет Boundaries, класи OrderMgr і TransactionMgr на пакет Control і клас Order на пакет Entities.

Додаймо діаграму класів до кожного пакету:

1) Двічі клацніть лівою кнопкою миші на пакеті Boundaries у вікні діаграм.
2) У вікні виберіть Так.
3) У вкладці Властивості, введіть ім'я для нової діаграми – MainB.
4) Клацніть на діаграму, щоб відкрити її.
5) В браузері виділіть клас OrderOptions (потім OrderDetail), клацніть ПКМ і виберіть Додати в діаграму, а потім натисніть мишею в поле діаграми.
6) Класи будуть відображатися на діаграмі.
7) Виконайте ці ж кроки, щоб помістити класи OrderMgr і TransactionMgr на головну діаграму класів пакета Control (MainC).
8) Виконайте ці ж дії, щоб помістити клас Order на головну діаграму класів пакета Entities (MainE).

Для вказівки відносин між класами знадобляться наступні кнопки:



2. Завдання до лабораторної роботи 5

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.

Лабораторна робота 6. Дослідження структури та умов використання методології UML та діаграми варіантів використання

Мета роботи:

Розглянути визначення бізнес-процесу та спосіб його опису за допомогою методології UML. Навчитися розробляти задачі у вигляді діаграми варіантів використання, описаної за методологією UML.

1. Теоретичний матеріал

Діаграма Use Case мови UML визначає поведінку програмної системи з погляду користувача. Діаграма Use Case розглядається як головний засіб для первинного моделювання динаміки системи, використовується для з'ясування вимог до системи, що розробляється, фіксації цих вимог у формі, яка дозволить проводити подальшу розробку. У літературі діаграми Use Case часто називають діаграмами прецедентів або діаграмами варіантів використання.

Основні графічні елементи діаграми варіантів використання:

– Варіант використання (use case)
– Актор (actor)
– Суб'єкт (subject)
– Коментар (comment)

Варіант використання (use case) являє собою загальну специфікацію сукупності виконуваних системою дій з метою надання деякого результату, що спостерігається, який має значення для одного або декількох акторів.

Інакше кажучи, варіант використання є закінченим фрагментом поведінки системи з погляду тих чи інших зацікавлених осіб. Хоча кожен варіант використання передбачає реалізацію системою деякої послідовності дій, які мають бути виконані з метою надання відповідного результату або сервісу, самі ці дії на діаграмі варіантів використання не зображуються.

Загальноприйнятим графічним зображенням варіанта використання в нотації мови UML є еліпс, всередині якого нижче записується ім'я у формі рядка тексту. Приклад варіанта використання представлений на рисунку 6.1.

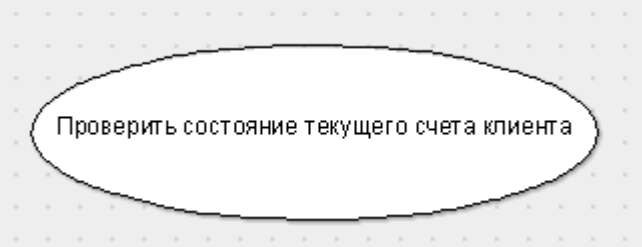


Рисунок 6.1 – Стандартне графічне позначення варіанта використання

Кожен варіант використання повинен мати ім'я, яке відрізняє його від інших. Ім'я варіанта використання являє собою текстовий рядок, який, узятий сам по собі, називається простим ім'ям. До кваліфікованого імені додається префікс – ім'я пакета, в якому знаходиться варіант використання, наприклад, «Датчики: Відкалібрувати положення».

Будь-який варіант використання має виконувати певний обсяг роботи. З погляду дійової особи він робить щось таке, що представляє певну цінність: наприклад, обчислює результат, створює новий об'єкт або змінює стан іншого об'єкта.

Актор (actor) являє собою зовнішню по відношенню до проєктованої системи сутність, яка взаємодіє з системою та використовує її функціональні можливості для досягнення певних цілей або вирішення приватних завдань.

Загальноприйнятим графічним зображенням актора на діаграмах у нотації мови UML є фігурка «дротяного чоловічка», під якою записується обов'язкове ім'я актора (рисунок 6.2).

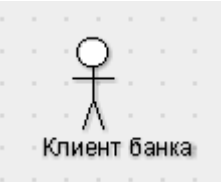


Рисунок 6.2 – Символ графічного позначення актору

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

Суб'єктом (subject) у контексті мови UML називається будь-який елемент моделі, який має функціональну поведінку. У літературі часто називається межею системи.

Суб'єкт варіантів використання може бути деякою фізичною системою або будь-яким іншим елементом із поведінкою, таким як підсистема, клас або компонент. Графічно суб'єкт на діаграмі варіантів використання зображується у формі прямокутника, всередині якого розміщуються окремі варіанти використання. Кожен суб'єкт має ім'я, яке має відповідати імені елемента системи, що моделюється, що володіє даною поведінкою.

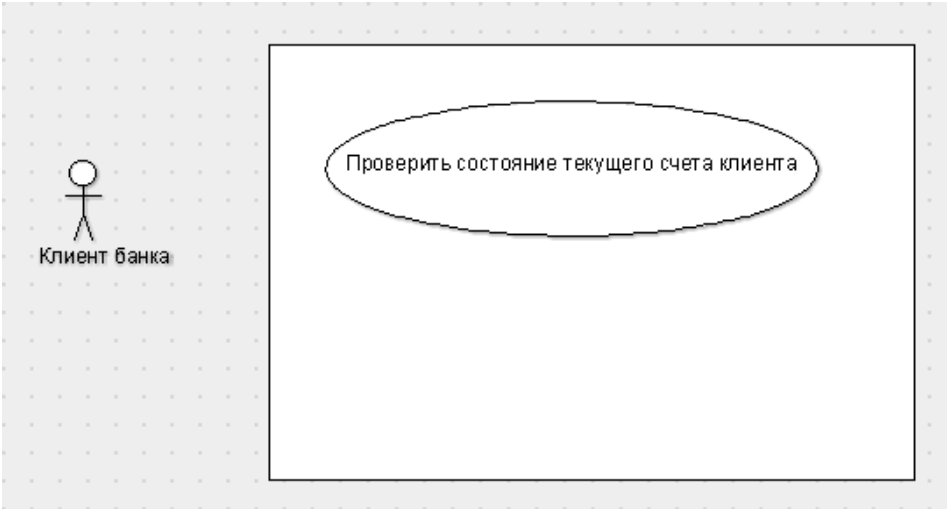


Рисунок 6.3 – Графічне позначення суб'єкта чи межі системи

Коментар (comment) в мові UML призначений для включення в модель довільної текстової у формі примітки, яка може бути приєднана до одного або кількох елементів моделі, що розробляється. Як така інформація може бути, наприклад, пояснення розробника щодо призначення елементів діаграми, рекомендації керівника проєкту з реалізації моделі, довідкова інформація про автора та особливості розробки окремих елементів моделі. Стосовно діаграм варіантів використання коментар може мати уточнюючу інформацію, що стосується контексту тих чи інших варіантів використання або акторів.

Графічно коментар на всіх типах діаграм зображується у формі прямокутника із «загнутим» правим верхнім куточком.

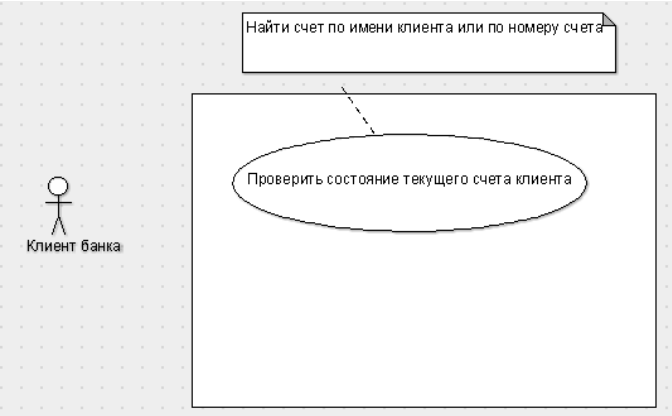


Рисунок 6.4 – Графічне позначення коментарів

Під *відношенням* (relationship) у мові UML розуміється довільний семантичний взаємозв'язок між окремими елементами моделі.

Види відносин на діаграмах варіантів використання:

– Асоціація (association)
– Відношення залежності (dependency)
– Відношення узагальнення (generalization)

Асоціація є одним із фундаментальних понять у мові UML і може використовуватися на різних канонічних діаграмах при побудові візуальних моделей. Стосовно діаграм варіантів використання, відношення асоціації може служити лише для позначення взаємодії актора з варіантом використання.

На діаграмах варіантів використання, як і на інших діаграмах, відношення асоціації позначається суцільною лінією, що з'єднує актора і варіант використання. При цьому слід розрізняти спрямовані (рисунок 6.5а) та неспрямовані (рисунок 6.5б) асоціації.



а б

Рисунок 6.5 – Приклад графічного уявлення відносини між актором та варіантом використання: а – спрямована асоціація; б – ненаправлена асоціація

У контексті діаграми варіантів використання спрямована асоціація від актора до варіанту використання може вказувати на те, що даний актор є ініціатором виконання відповідного варіанту використання. Спрямована асоціація від варіанта використання до актора може вказувати також на той факт, що даному актору надається довідкова або звітна інформація про результати функціонування системи, що моделюється..

Відношення залежності (dependency) визначається як форма взаємозв'язку між двома елементами моделі, призначена для специфікації тієї обставини, що зміна одного елемента моделі призводить до зміни деякого іншого елемента.

Відношення включення (include) є окремим випадком відносини залежності і специфікує той факт, що деякий варіант використання містить поведінку, визначену в іншому варіанті використання.

Графічно це відношення позначається як відношення залежності у формі пунктирної лінії з «V» – образною стрілкою, спрямованою від залежного варіанта використання до незалежного варіанта використання (рисунок 6.6). Залежний варіант використання часто називають також базовим або таким, що включає, а незалежний – варіантом використання, що включається. При цьому лінія зі стрілкою обов'язково позначається ключовим словом <<include>>, записаним у формі

стереотипу.

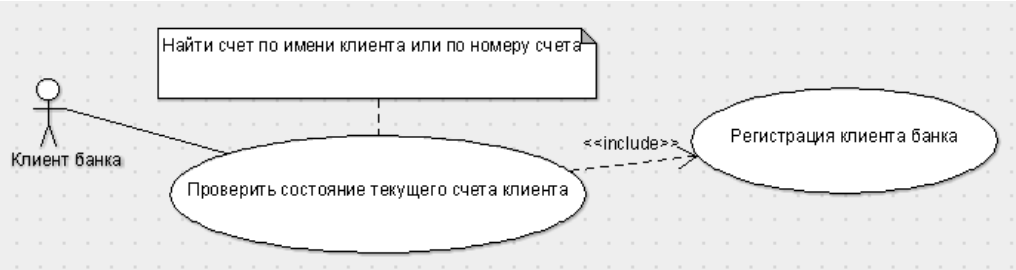


Рисунок 6.6 – Приклад графічного зображення відношення включення між варіантами використання

Відношення розширення (extend) є окремим випадком відношення залежності і визначає взаємозв'язок одного варіанту використання з деяким іншим варіантом використання, функціональність або поведінка якого задіюється першим не завжди, а лише при виконанні деяких додаткових умов.

Графічно дане відношення позначається як відношення залежності у формі пунктирної лінії з «V»-образною стрілкою, спрямованою від залежного варіанту використання до незалежного варіанту використання та з'єднаною з ним у так званій точці розширення (extension point) (рисунок 6.7). Залежний варіант використання часто називають також розширюючим, а незалежний – базовим або варіантом використання, що розширюється. При цьому лінія зі стрілкою обов'язково позначається ключовим словом <<extend>>, записаним у формі стереотипу.

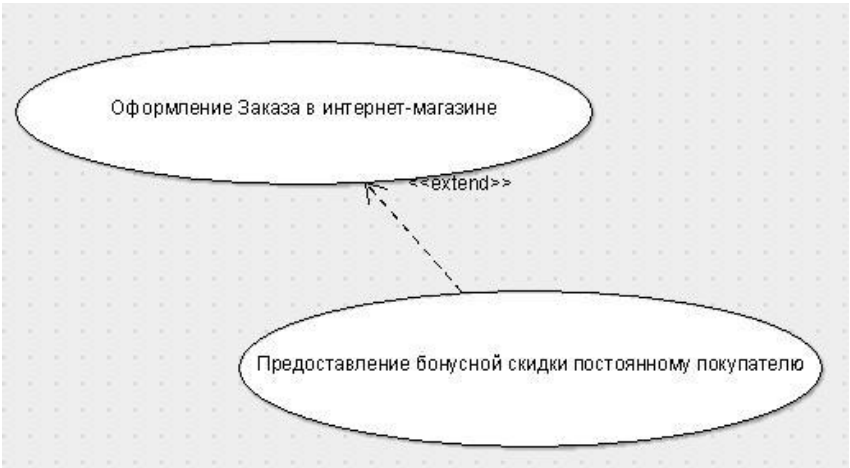


Рисунок 6.7 – Приклад графічного зображення відношення розширення між варіантами використання

Розширюваний варіант використання може мати окремий розділ, в якому може бути описана точка розширення, а умови розширення можуть бути наведені в коментарі з ключовим словом Condition (рисунок 6.8).

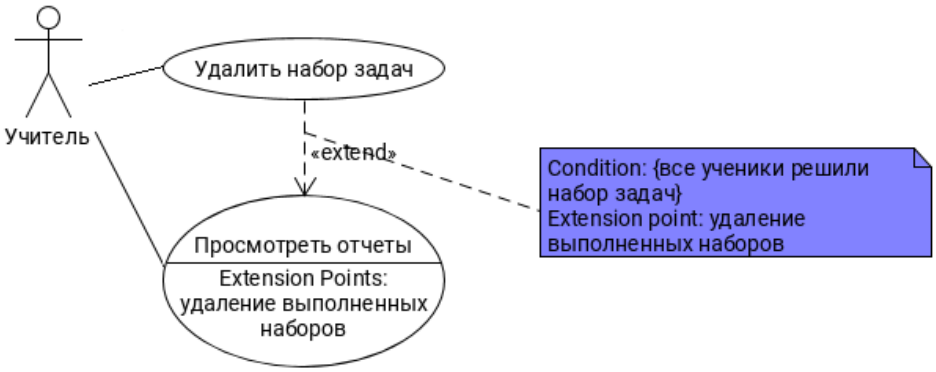


Рисунок 6.8 – Приклад використання точки розширення

Відношення узагальнення (generalization) призначене для специфікації того факту, що один елемент моделі є спеціальним або окремим випадком іншого елемента моделі.

Графічно відношення узагальнення позначається суцільною лінією зі стрілкою у формі не зафарбованого трикутника, що вказує на загальний елемент моделі.

У зображеному на рисунку 6.9 фрагменті діаграми варіантів використання ставлення узагальнення специфікує те, що варіант використання «Оплата товару за кредитною карткою» є спеціальним випадком варіанту використання «Оплата обраного в інтернет-магазині товару». При цьому варіант використання «Оплата обраного в інтернет-магазині товару», який є нащадком, успадковує всі властивості поведінки батька, тобто, варіанти використання «Оплата товару за кредитною карткою».

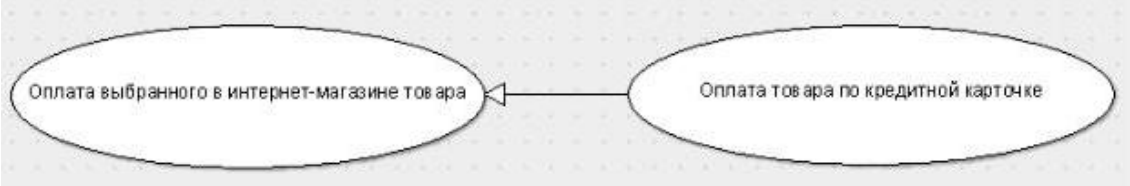


Рисунок 6.9 – Приклад графічного зображення відношення узагальнення між варіантами використання

Приклад графічного зображення відношення узагальнення між акторами показано на рисунку 6.10.



Рисунок 6.10 – Приклад графічного зображення відношення узагальнення між акторами

Вимога (requirement) – бажана властивість, характеристика чи умова, якою має задовольняти програмна система у процесі своєї експлуатації.

Специфікація вимог до проєктної системи у формі діаграми варіантів використання та додаткових сценаріїв може бути самостійною моделлю, яка в мові UML отримала назву моделі варіантів використання.

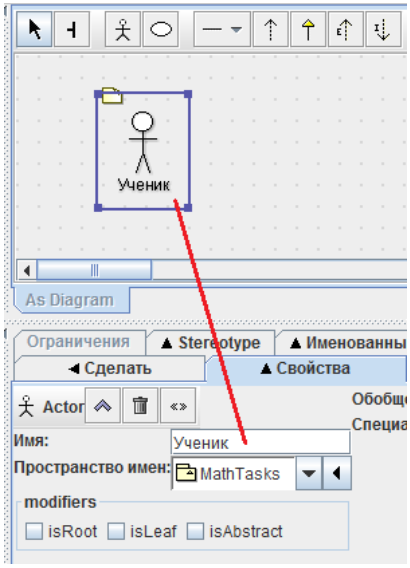
Зміст варіанта використання може бути додатково специфікований у формі пояснювального тексту, який розкриває зміст або семантику дій, що виконуються при виконанні варіанта використання. Такий пояснювальний текст стосовно діаграм варіантів використання отримав спеціальну назву тексту-сценарію або просто – сценарію.

Сценарій (scenario) – спеціально написаний текст, який описує поведінку моделі, що моделюється, у формі послідовності виконуваних дій акторів і самої системи.

Розглянемо основні кроки, які необхідно виконати в програмі ArgoUML.

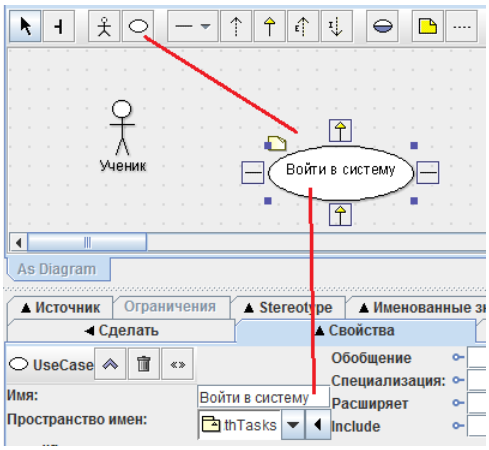
Щоб створити діаграму варіантів використання, на панелі інструментів виберіть піктограму «Діаграма варіантів використання» ().

Створимо актора. Виберемо піктограму Новий актор (). Потім лівою кнопкою миші натисніть по області, в якій буде зображена діаграма. На ній з'явиться новий актор. Встановимо для актора ім'я. Виділяємо актора та скористаємося вкладкою Властивості та полем Ім'я.



Аналогічно створюємо актора Вчитель.

Учень може «Увійти до системи» та «Виконати набір завдань». Це будуть варіанти використання учня. Створимо варіант використання. Виберемо піктограму «Новий варіант використання» (). Потім лівою кнопкою миші натисніть по області, в якій буде зображена діаграма. На ній з'явиться новий варіант використання. Введемо для варіанту використання ім'я. Це можна зробити у вкладці Властивості у полі Ім'я.



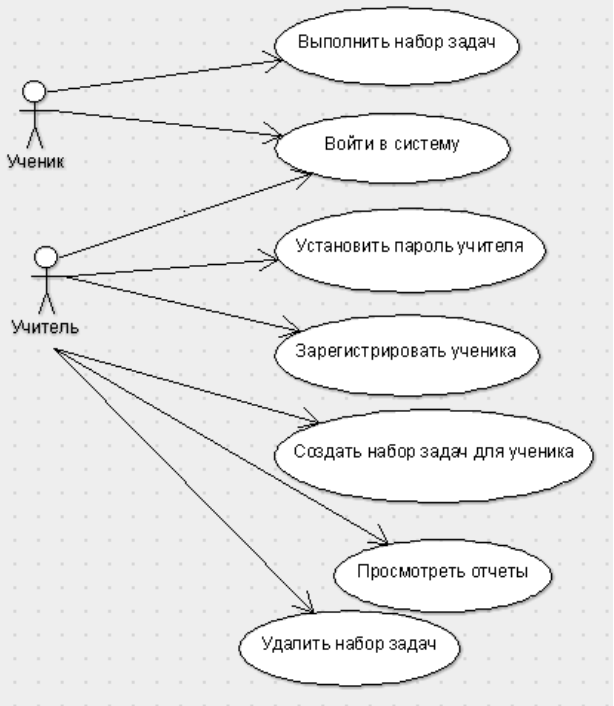
Аналогічно додається варіант використання «Виконати набір завдань».

Встановимо відношення асоціації між створеним актором та варіантом використання. Виберемо піктограму «Нова асоціація» (). Проведемо її із затиснутою лівою кнопкою миші від актора до варіанта використання.



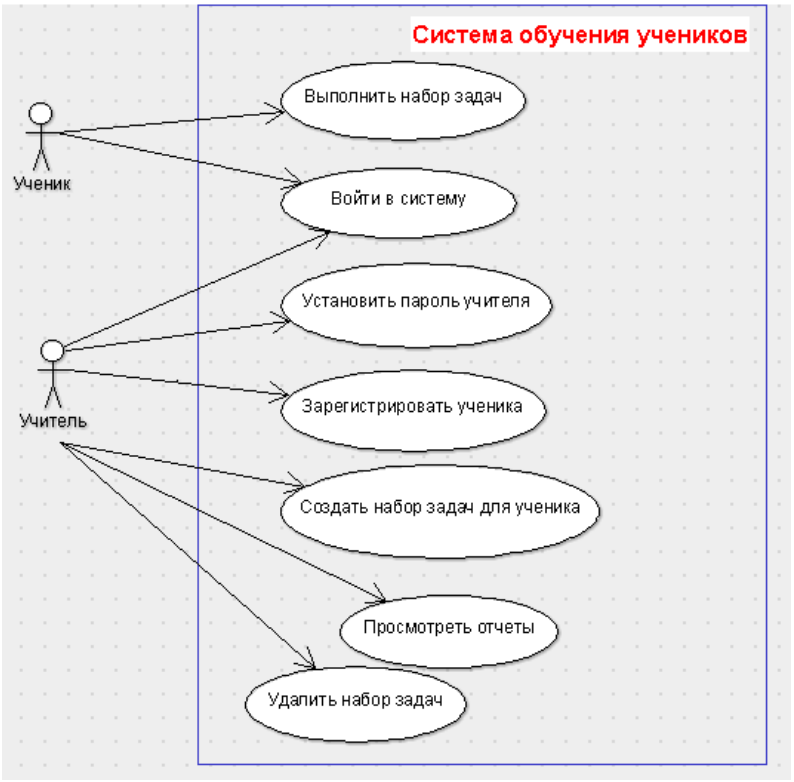
Аналогічно для актора «Вчитель» створюються варіанти використання «Встановити пароль вчителя», «Зареєструвати учня», «Створити набір завдань для учня», «Переглянути звіти»,

«Видалити набір завдань». Аналогічно встановити ставлення асоціації між «Вчителем» та його варіантами використання та додати ставлення асоціації до варіанту використання «Увійти в систему».



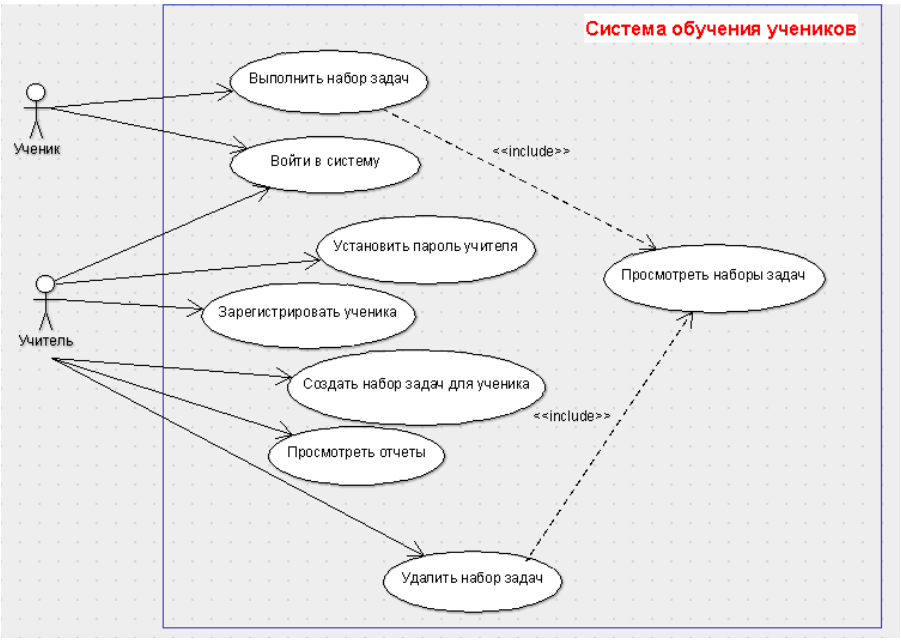
Укладемо створені варіанти використання суб'єкта. Для цього виберіть піктограму «Прямокутник» (). Помістимо прямокутник на ділянці діаграми. Змінимо розміри прямокутника так, щоб він охоплював усі варіанти використання. Також у прямокутнику у властивості «Заповнити» на вкладці «Стиль» встановимо значення «Не заповнювати».

Щоб створити ім'я суб'єкта, виберіть піктограму «Текст» (). Помістимо текст на ділянку діаграми. У вкладці «Стиль» для розділу «Лінія» встановимо значення «Не лінія». Подвійне клацання лівої кнопки миші по елементу «Текст для введення тексту».



Відношення включення вказує на те, що поведінка одного прецеденту включається в деякій точці в інший прецедент як складовий компонент. Особливості включення полягають у тому, що включений прецедент може бути обов'язковим для доповнюваного, тобто це ставлення задає дуже сильний зв'язок. Наприклад, якщо ми хочемо зобразити на діаграмі той факт, що «Видалення набору завдань» вчителем та «Виконання завдань» учнем не повинно відбуватися без обов'язкового «Перегляду наборів завдань», то нам потрібно використовувати ставлення включення:

Щоб з'єднати два варіанти використання ставленням включення, виберіть піктограму «New include» (). Проведемо її із затиснутою лівою кнопкою миші від одного варіанту використання до іншого.

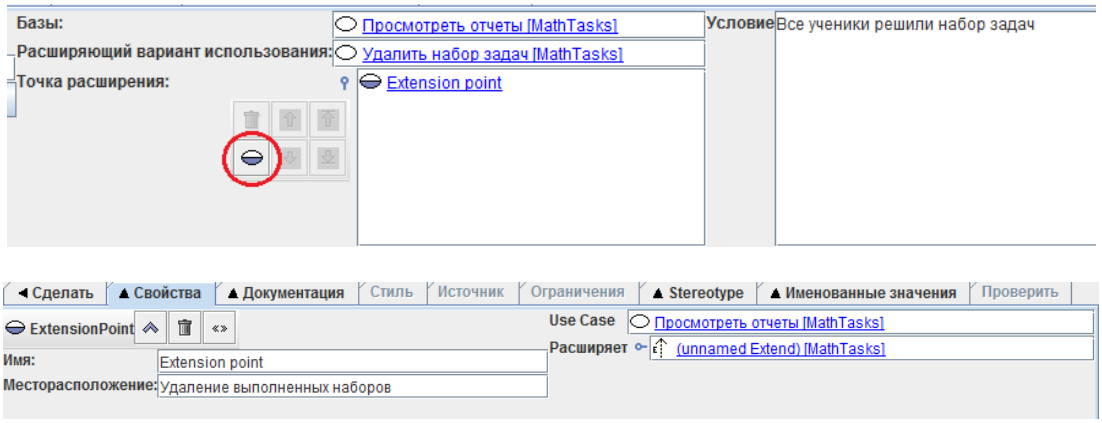


Відношення розширення відображає можливе приєднання одного варіанта використання до іншого в певній точці (точці розширення). При цьому наголошується на тому, що розширюючий варіант використання виконується лише за певних умов і не є обов'язковим для виконання

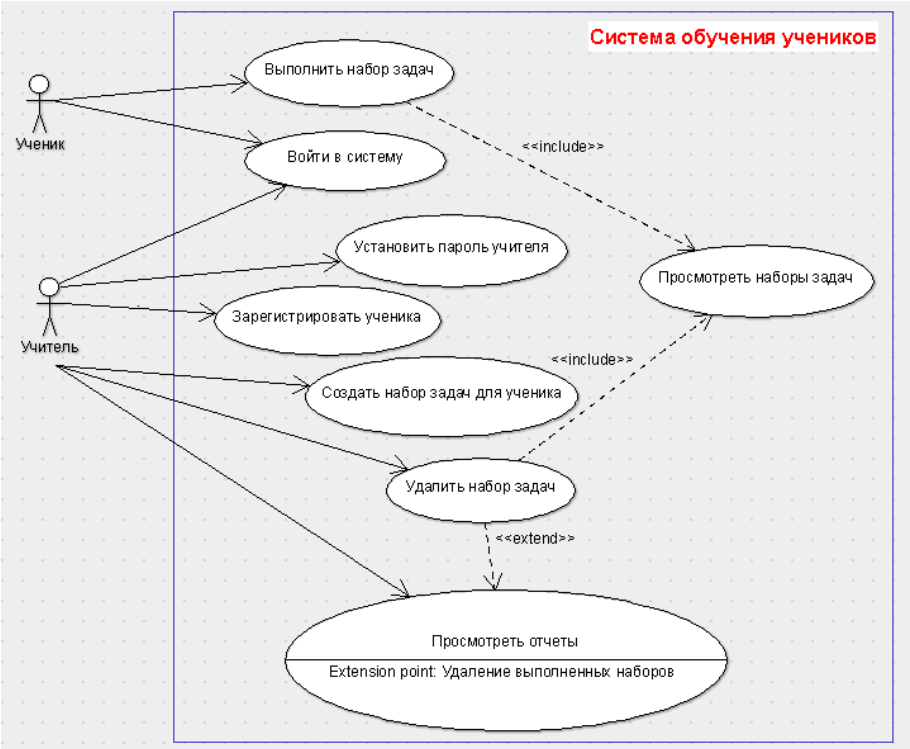
основного прецеденту. На діаграмі такий вид відношення зображується стрілкою, спрямованою до прецеденту, що розширюється, в окремому розділі якого може бути описана точка розширення, а умови розширення можуть бути наведені в коментарі з ключовим словом Condition.

Наприклад, відношення розширення слід застосувати, якщо за технічним завданням потрібна можливість видалення набору завдань у варіанті використання «Перегляд звітів» за умови, що всі учні вирішили цей набір.

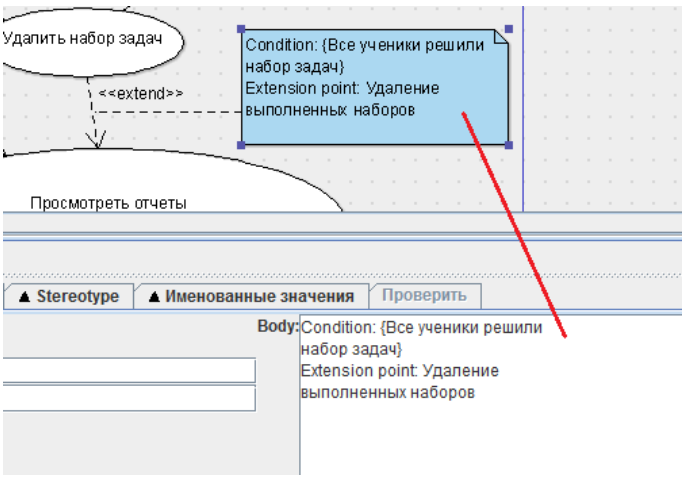
Щоб поєднати два варіанти використання ставленням розширення, виберемо піктограму «New extend» (). Проведемо її із затиснутою лівою кнопкою миші від одного варіанту використання до іншого. У властивостях відношення вказуються «Точка розширення» та «Умова».



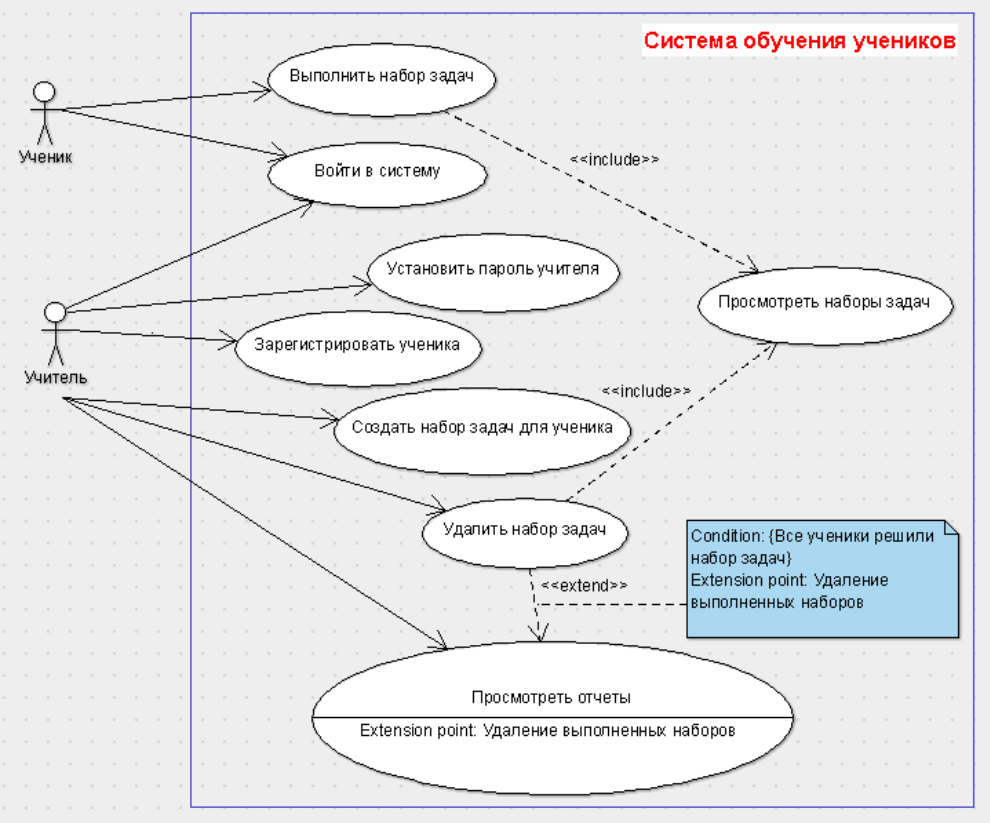
Щоб у варіанті використання показувалася точка розширення, виділіть її та в закладці «Стиль» поставте галочку «Точки розширення».



Оскільки умова точки розширення не відображається на діаграмі, її можна додати інформаційно у вигляді примітки. Для додавання примітки виділимо відношення розширення на діаграмі та виберемо піктограму «New comment» (). Суть коментаря додається до поля «Body».



В результаті отримуємо діаграму «Варіантів використання».



2. Завдання до лабораторної роботи 6

Оберіть задачу з області кібербезпеки, захисту інформації або іншої діяльності. Сформулювати обрану задачу. Створити діаграму варіантів використання або для сформульованої задачі або для функціонального опису деякого класу з лабораторної роботи 5.

Лабораторна робота 7. Дослідження структури та умов використання методології UML. Діаграма станів

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 з використанням діаграми станів.

1. Теоретичний матеріал

Модель стану представляє тимчасові, поведінкові та управлінські аспекти системи.

Кожна прикладна система характеризується як структурою складових її елементів, так і деякою поведінкою чи функціональністю. Для загального уявлення про функціональність моделі, що моделюється, призначені діаграми варіантів використання, які на концептуальному рівні описують поведінку системи в цілому. Зараз наше завдання полягає в тому, щоб подати поведінку більш детально на логічному рівні, тим самим розкрити сутність відповіді на питання: «У процесі якої поведінки система забезпечує необхідну функціональність?»

Для моделювання поведінки на логічному рівні у мові UML можуть використовуватися відразу кілька канонічних діаграм: станів, діяльності, послідовності та кооперації, кожна з яких фіксує увагу на окремому аспекті функціонування системи.

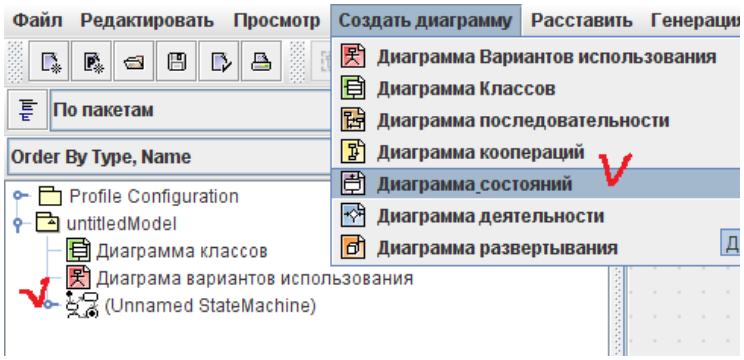
Діаграма станів описує процес зміни станів лише одного класу, а точніше – одного екземпляра певного класу, тобто моделює всі можливі зміни у стані конкретного об'єкта. При цьому зміна стану об'єкта може бути викликана зовнішніми впливами інших об'єктів або ззовні. Саме для опису реакції об'єкта на подібні зовнішні дії використовуються діаграми станів.

Головне призначення цієї діаграми – описати можливі послідовності станів та переходів, які в сукупності характеризують поведінку елемента моделі протягом його життєвого циклу.

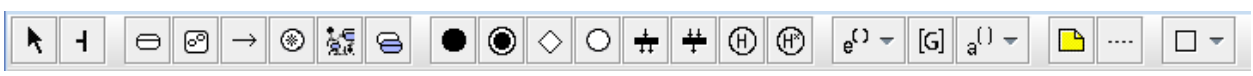
Реактивним називається такий об'єкт, поведінка якого найкраще характеризується його реакцією на події, які він отримує ззовні його контексту. Реактивний об'єкт має чіткий час життя, та його поточна поведінка залежить від минулого. Діаграми станів можуть бути приєднані до класів, варіантів використання або до всієї системи з метою візуалізації, специфікації, конструювання та документування динаміки окремих об'єктів.

Діаграма станів по суті є графом спеціального виду, який представляє певний автомат. Вершинами цього графа є стани та інші типи елементів автомата (псевдостани), які зображуються відповідними графічними символами. Дуги графа служать для позначення переходів зі стану до стану. Діаграми станів можуть бути вкладені одна в одну, утворюючи вкладені діаграми більш детального представлення окремих елементів моделі.

Для створення діаграми стану в ArgoUML необхідно в меню «Створити діаграму» вибрати пункт «Діаграма станів».



В результаті з'явиться панель вибору компонентів діаграми стану:



Компонентами діаграми станів є:

– Стани та підстани;

– Переходи.

Автомат (state machine) – це опис послідовності станів, якими проходить об'єкт протягом життєвого циклу, реагуючи на події, і навіть опис реакції цих подій.

Формалізм звичайного автомата ґрунтується на виконанні наступних обов'язкових умов:

– Автомат не запам'ятовує історію переміщення зі стану до стану;
– У кожний момент часу автомат може перебувати в одному і лише в одному зі своїх станів;
– Тривалість знаходження автомата у тому чи іншому стані і навіть час досягнення тієї чи іншої стани не специфікуються, тобто час на діаграмі станів присутній у неявному вигляді, хоча окремих подій може бути зазначений інтервал часу й у явному вигляді;
– Кількість станів автомата повинна бути обов'язково кінцевою (початковий та кінцевий стан);
– Кожен перехід повинен обов'язково з'єднувати два стани автомата. Допускається перехід зі стану у собі. Такий перехід ще називають «петлею»;
– Автомат не повинен містити конфліктуючих переходів, тобто таких переходів з того самого стану, коли об'єкт одночасно може перейти в два і більше наступних станів (крім випадку паралельних підавтоматів).

Формалізм автоматів допускає вкладення одних автоматів до інших. У цьому випадку вкладені автомати отримали назву підавтоматів.

Стан (state) – ситуація у життєвому циклі об'єкта, протягом якої він задовольняє певній умові, виконує деяку діяльність чи чекає певного події.

Стан на діаграмі зображується прямокутником із заокругленими вершинами . Цей прямокутник, своєю чергою, може бути поділений на дві секції горизонтальною лінією. Якщо вказана лише одна секція, то записується лише ім'я стану. В іншому випадку в першій записується ім'я стану, а в другій – список деяких внутрішніх дій або переходів в даному стані.

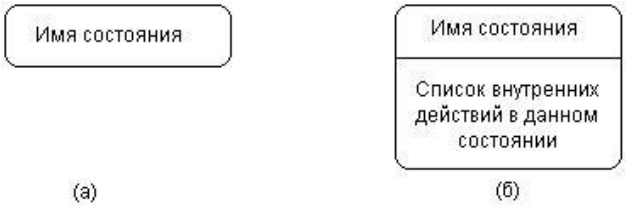


Рисунок 7.1 – Графічне зображення станів на діаграмі станів

Ім'я стану є рядком тексту, який розкриває змістовний зміст цього стану. Ім'я завжди записується з великої літери. Для ідентифікації імені стану рекомендується використовувати дієслова в даний час (дзвонить, друкує, чекає) або відповідні дієприкметники (зайнятий, вільний, переданий, отриманий). Ім'я стану може бути відсутнім. І тут стан є анонімним, і якщо діаграмі станів їх кілька, всі вони повинні різнитися між собою.

Розглянемо перелік внутрішніх дій або діяльностей, які виконуються в процесі знаходження елемента, що моделюється в даному стані. Кожна з дій записується у вигляді окремого рядка та має наступний формат:

<мітка-дії '/' вираз-дії>

Мітка дії вказує на обставини або умови, за яких виконуватиметься діяльність, визначена виразом дії. Якщо список виразів дії порожній, то роздільник, як похила риса '/', може вказуватися.

Перелік міток дії має фіксовані значення в мові UML, які не можуть бути використані як імена подій. Ці значення такі:

– entry – ця мітка вказує на дію, специфіковану наступним за нею виразом дії, що виконується в момент входу в даний стан (вхідна дія);
– exit – ця мітка вказує на дію, специфіковану наступним за нею виразом дії, яка виконується в момент виходу з цього стану (вихідна дія);
– do – ця мітка специфікує діяльність, що виконується ("do activity"), яка виконується протягом усього часу, поки об'єкт знаходиться в даному стані, або до тих пір, поки не

закінчиться обчислення, специфіковане наступним за нею виразом дії. В останньому випадку після завершення події генерується відповідний результат;

– include – ця мітка використовується для звернення до підавтомату, при цьому наступний вираз дії містить ім'я цього підавтомата.

У решті випадків мітка дії ідентифікує подію, яка запускає відповідний вираз дії. Ці події називаються *внутрішніми переходами* і семантично еквівалентні переходам у саме цей стан.

Розглянемо приклад стану в ситуації введення пароля користувача під час аутентифікації входу в деяку програмну систему. У цьому випадку список внутрішніх дій у даному стані не порожній і включає 4 окремі дії, перші дві з яких стандартні та описані вище, а два останні визначаються своєю специфікацією.

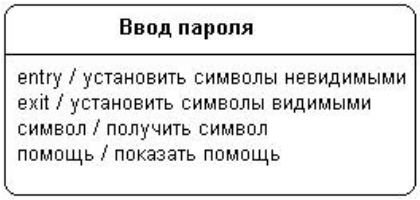
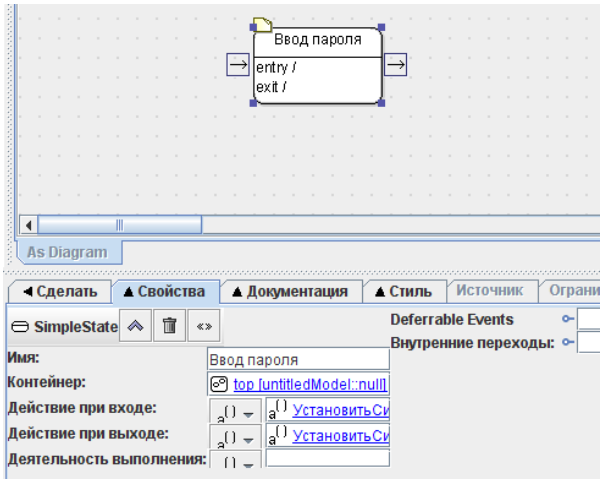


Рисунок 7.2 – Приклад стану із секцією внутрішніх дій

Приклад стану в ArgoUML:



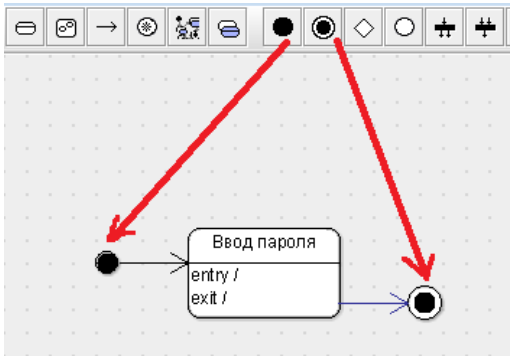
Початковий стан є окремим випадком стану, який не містить ніяких внутрішніх дій. У цьому стані знаходиться об'єкт за замовчуванням у початковий момент часу. Графічно початковий стан у мові UML позначається зафарбованим кружечком, з якого може лише виходити стрілка, відповідна переходу.

Кінцевий (фінальний) стан є окремим випадком стану, який також не містить ніяких внутрішніх дій. У цьому стані буде об'єкт за замовчуванням після завершення роботи автомата в кінцевий момент часу. Графічно кінцевий стан у мові UML позначається зафарбованим кружечком, поміщеним у коло, у якому може лише входити стрілка, відповідна переходу.



Рисунок 7.3 – Приклад початкового та кінцевого стану

Приклад початкового та кінцевого стану в ArgoUML:



Перехід (transition) – зв'язок між двома станами, що показує, що об'єкт, що у першому стані, має виконати деякі дії і перейти у друге, щойно станеться певна подія і будуть виконані певні умови.

Перехід здійснюється при настанні деякої події: закінченні виконання (do activity), отриманні об'єктом повідомлення або прийомі сигналу. На переході вказується ім'я події, і можуть вказуватися дії, що проводяться об'єктом у відповідь на зовнішні події при переході з одного стану в інший. Спрацювання переходу може залежати не тільки від настання певної події, а й від виконання певної умови, яка називається сторожовою умовою. Об'єкт перейде з одного стану в інший у тому випадку, якщо відбулася вказана подія та сторожова умова набула значення «істина».

На діаграмі станів перехід зображується суцільною лінією зі стрілкою, яка спрямована на цільовий стан.

Кожен перехід може бути позначений рядком тексту, який має наступний загальний формат:

<сигнатура події>'['<сторожова умова>']' <вираз дії>.

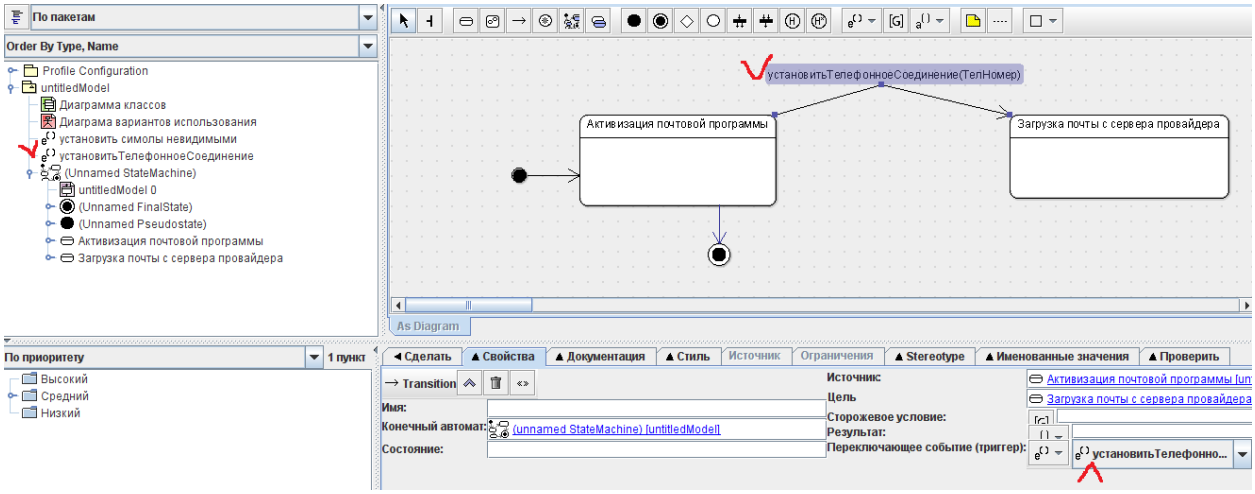
При цьому сигнатура події описує певну подію з необхідними аргументами:

<ім'я події>'('<список параметрів, розділених комами>')'.

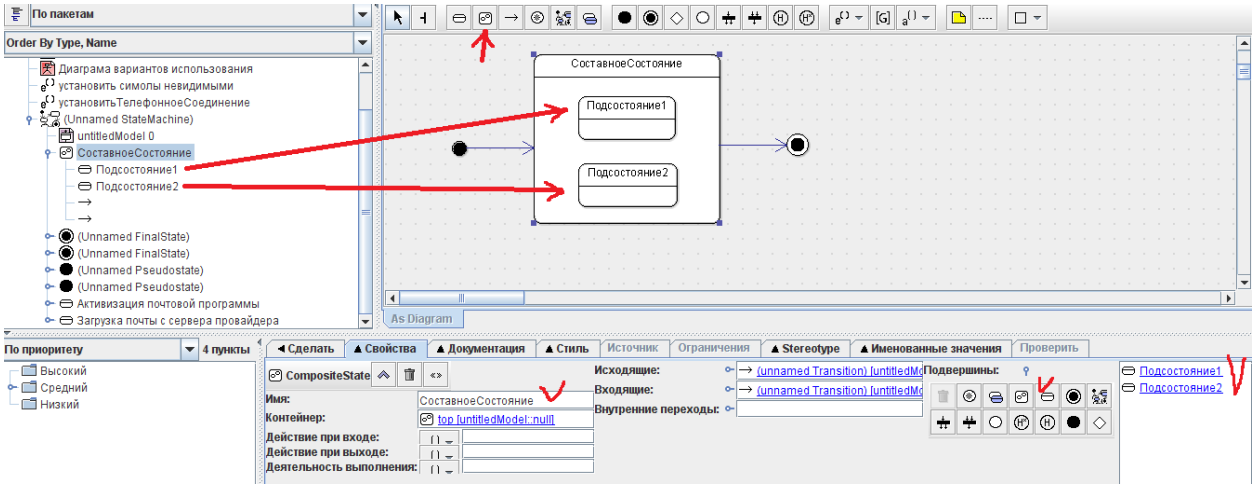
Подія (event) – специфікація істотного факту, що відбувається у часі та просторі. У контексті автомата подія – це вплив, що викликає перехід між станами.

У мові UML події грають роль стимулів, які ініціюють переходи з одних станів до інших. Як події можна розглядати сигнали, дзвінки, закінчення фіксованих проміжків часу або моменти закінчення виконання певних дій. Ім'я події ідентифікує кожен окремий перехід на діаграмі станів і може містити рядок тексту, що починається з малої літери.

Сторожова умова (guard condition), якщо вона є, завжди записується в прямих дужках і є деяким булевським виразом. Якщо сторожова умова набуває значення «істина», то відповідний перехід може спрацювати. Якщо ж сторожова умова набуває значення «брехня», то перехід не може спрацювати, і за відсутності інших переходів об'єкт не може перейти в цільовий стан цього переходу.

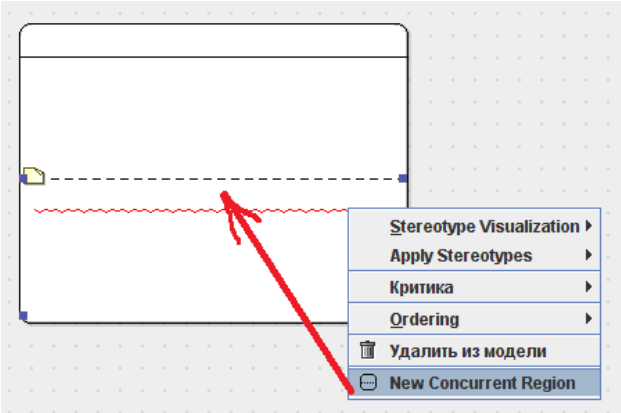


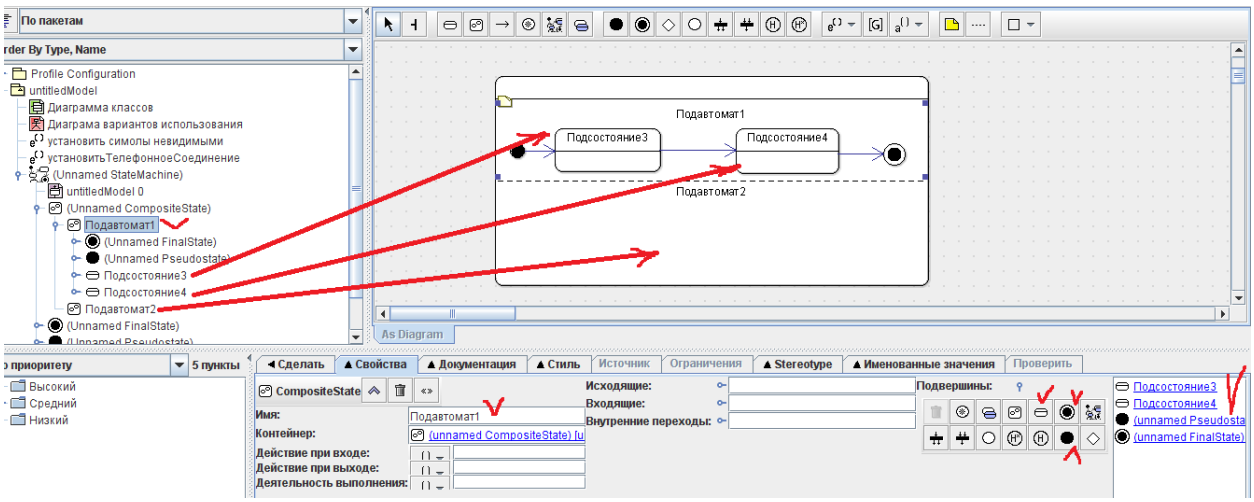
Складовий стан (composite state) – такий складний стан, що складається з інших вкладених у нього станів. Останні виступатимуть по відношенню до першого як субстани (substate). Хоча між ними має місце композиція, графічно всі вершини діаграми, які відповідають вкладеним станам, зображуються всередині символу складового стану.



Послідовні підстани (sequential substates) використовуються для моделювання такої поведінки об'єкта, під час якої в кожний момент часу об'єкт може перебувати в одному і лише одному підстані. Поведінка об'єкта в цьому випадку є послідовною зміною підходів, починаючи від початкового і закінчуючи кінцевим підстанами.

Паралельні стани (concurrent substates) дозволяють специфікувати два і більше підавтоматів, які можуть виконуватися паралельно всередині складової події. Кожен із підавтоматів займає деяку область (регіон) усередині складового стану, яка відокремлюється від інших горизонтальною пунктирною лінією. Якщо на діаграмі станів є складовий стан із вкладеними паралельними підстанами, то об'єкт може одночасно перебувати в кожному з цих підстанів.





Діяльність (activity) специфікує роботу, що відбувається усередині автомата.

Дія (action) – примітивне обчислення, що призводить до зміни стану моделі або повернення значення.

2. Завдання до лабораторної роботи 7

Побудувати діаграми станів для розроблених варіантів використання.

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.

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.

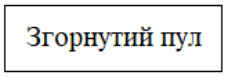
Пул являє собою:

– 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:



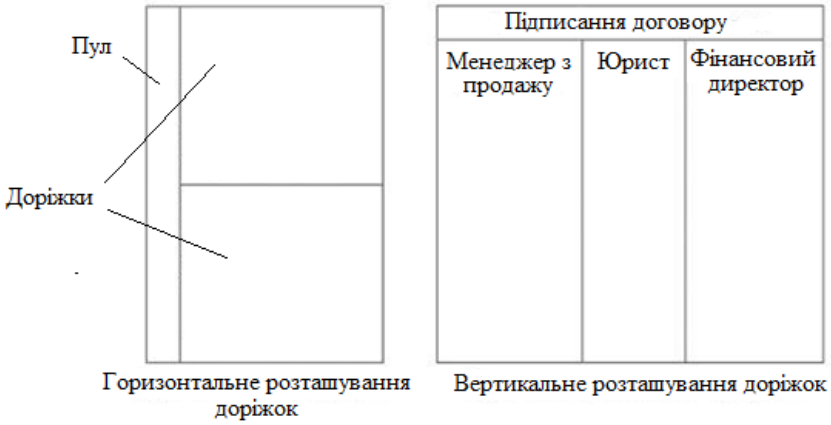
Згорнутий пул – елемент, що позначає зовнішній (по відношенню до поточної діаграмі) процес або зовнішнє посилання. Усередині блоку поміщається найменування зовнішнього процесу або зовнішнього посилання.



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:

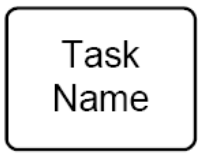


У кожній доріжці розташовуються дії, що виконуються одним виконавцем.

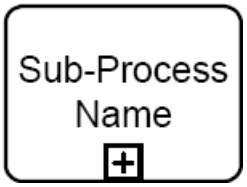
Під *Дією* розуміється одиниця роботи, що виконується в ході виконання бізнес-процесу. Дії можуть бути як елементарними (завдання / task), так і складовими (підпроцес / sub-process).

BPMN передбачає наступні графічні відображення для основних типів дій:

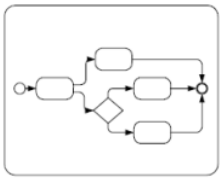
Абстрактна задача – використовується для позначення простого дії або операції, яка не має подальшої декомпозиції в рамках поточного бізнес-процесу.



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.



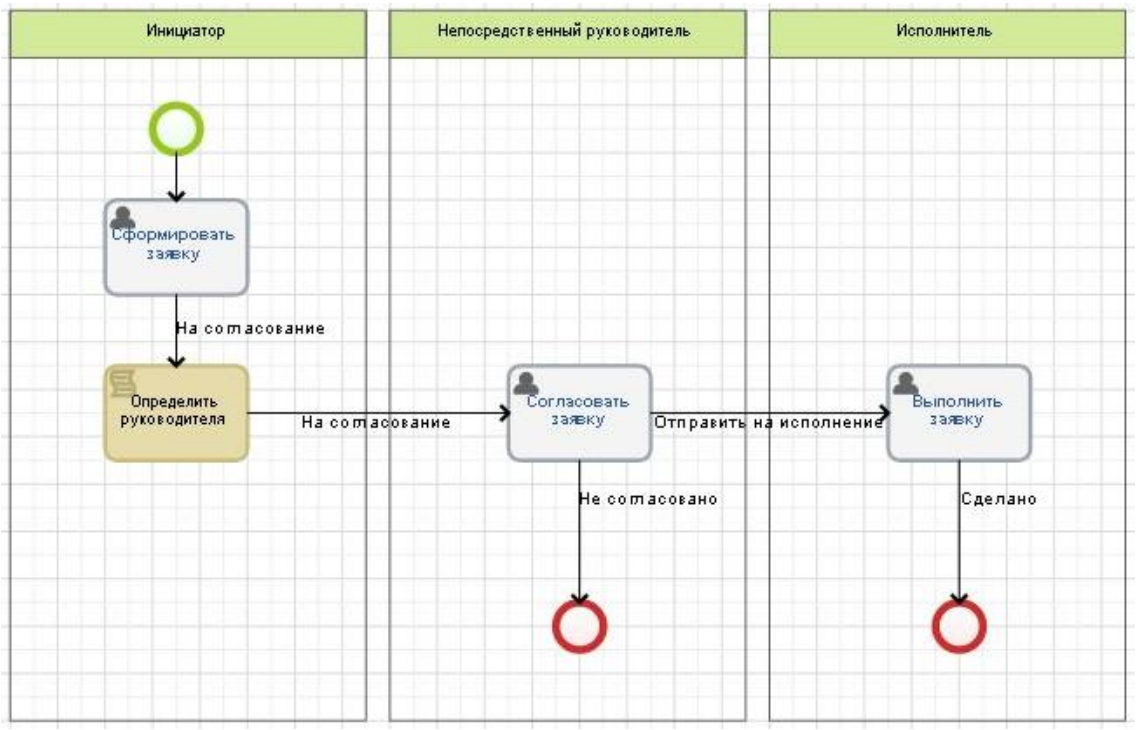
Події-посилання (link events) використовуються як міжсторінкових з'єднання. Пара відповідних посилань еквівалентна потоку управління.



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

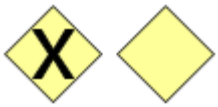


Наприклад, продаж та відвантаження обладнання:

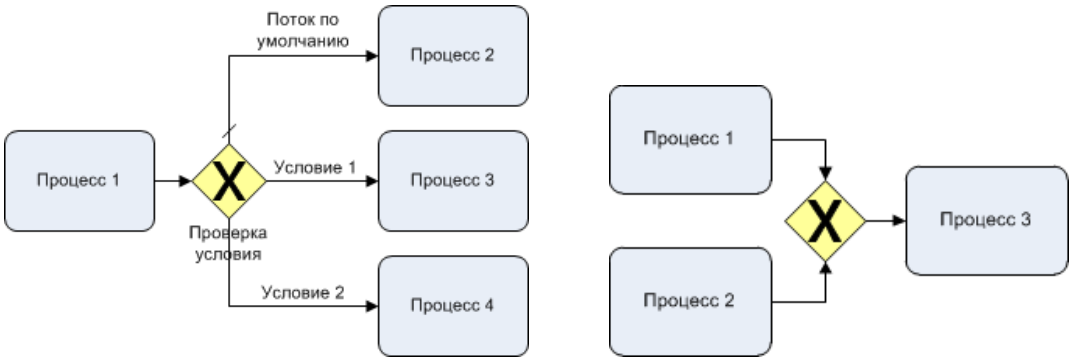


Під *шлюзами* розуміються елементи, що визначають розгалуження та злиття потоків робіт. Шлюз можна уявити як пропускний пристрій, який або пропускає потік, або ні.

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.



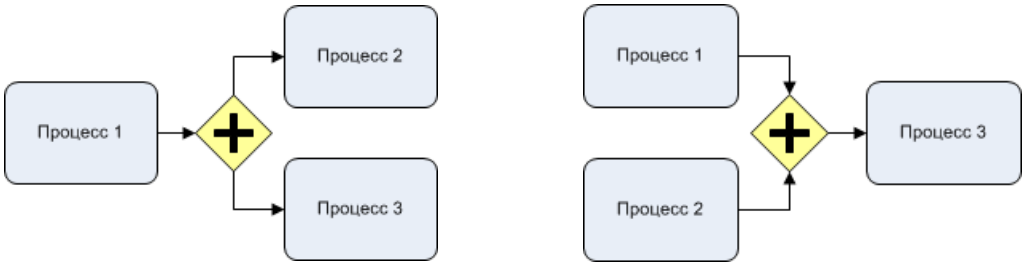
Наприклад:



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.

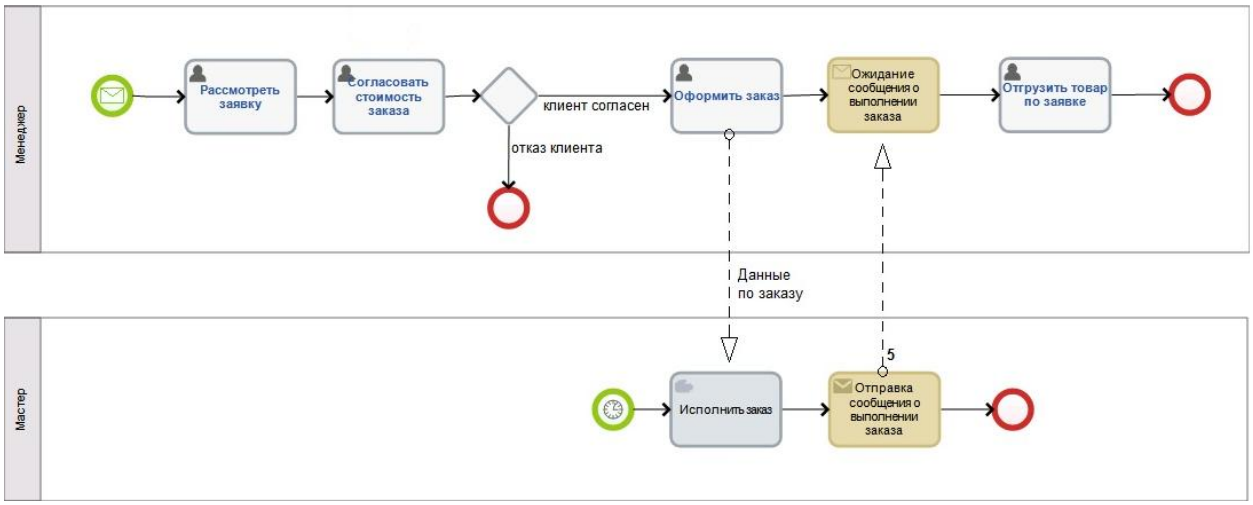


Наприклад:



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:

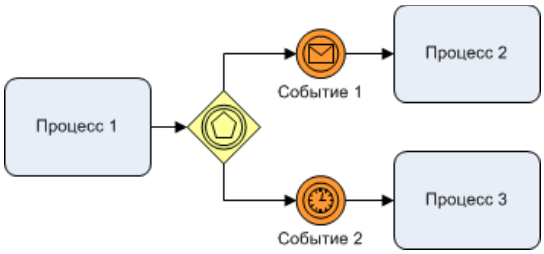


Залежність від події – розвилка з залежністю від події схожа на розвилку «виключення». Однак в разі залежності від події напрямок диктується тим, яка була вчинена подія, а не тим, яка була виконана умова.



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.

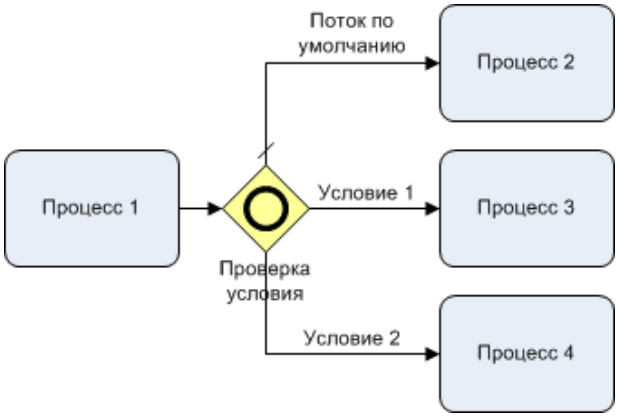


Включення – розбиває потік процесу на одне або кілька напрямків. Наприклад, розвилка «включення» може вказувати на дії, вжиті компанією в результаті отриманих результатів опитування.



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.

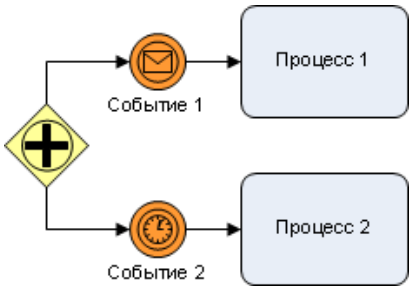
Наприклад, якщо Умова 1 вірно, то виконається Процес 3; якщо Умова 2 вірно, то виконається Процес 4; якщо ні Умова 1, ні Умови 2 неправильні, то виконається тільки Процес 2.



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.



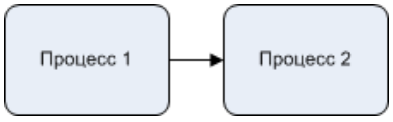
Приклад позначення:



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

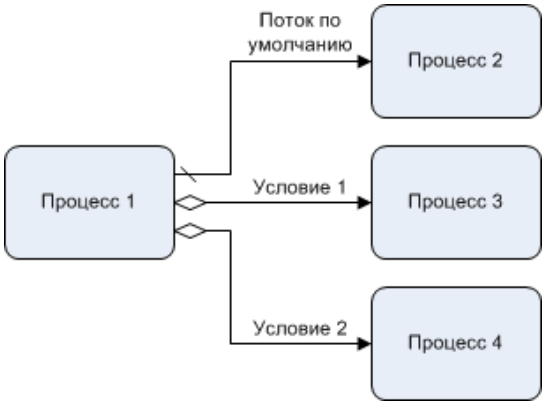
подальше виконання процесу буде відбуватися за певним потоком тільки якщо не виконується жодна з заданих умов.



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:



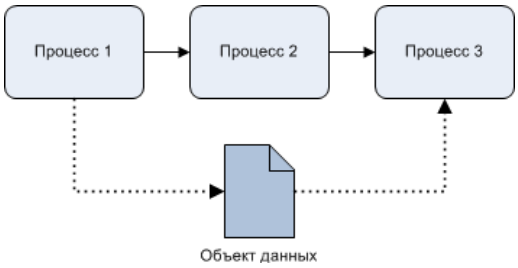
Ромбик біля основи стрілки додається, якщо умовний потік управління є вихідним від процесу. Ромбик не додають, якщо умовний потік управління є вихідним від шлюзу.

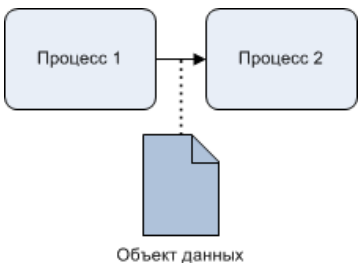
Потік повідомлень – використовується для відображення взаємодії між процесами - відображає передачу повідомлень або об'єктів з одного процесу в інший процес або зовнішнє посилання.



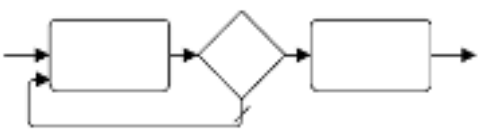
Асоціація – застосовується для візуалізації зв'язку між елементами потоку і об'єктами, які не є елементами потоку (артефактами).







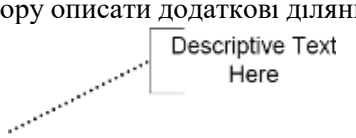
Цикли можуть з'являтися завдяки приєднанню Потоку операцій до «протилежного» об'єкту.



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.



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, шлюзів і артефактів за потреби 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.

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 (верхневрівнева 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).



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:

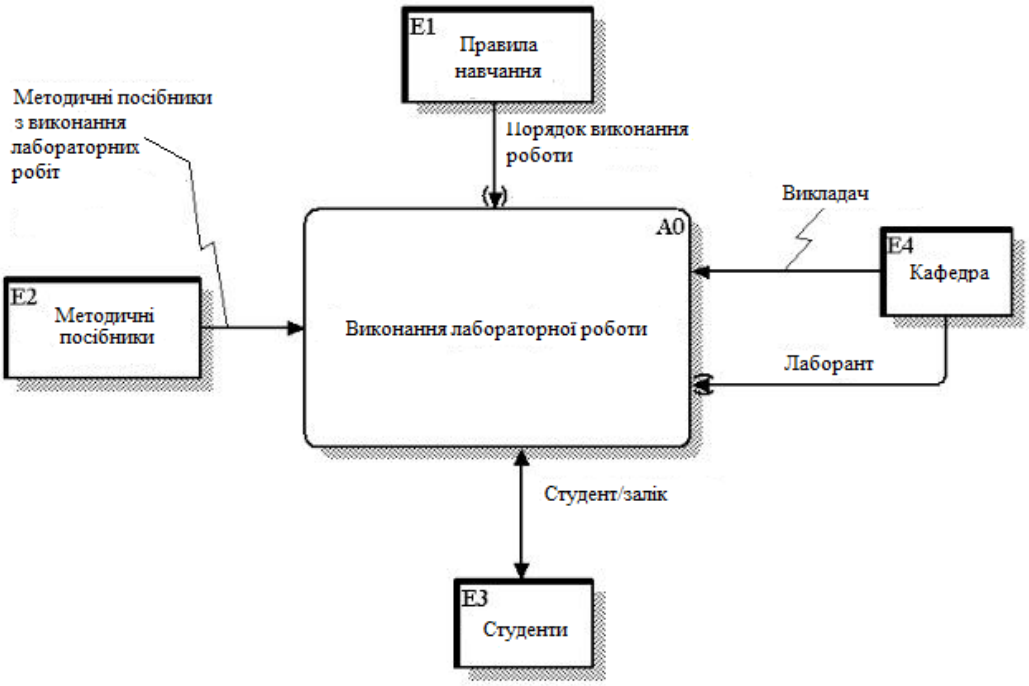


Рисунок 9.1 – Приклад процесу «Виконання лабораторної роботи»

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.

Лабораторна робота 10. Дослідження структури та умов використання методології EPC

Мета роботи:

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

1. Теоретичний матеріал

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

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

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

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

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

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

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

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

Нотація допускає проведення декомпозиції на більш низьких рівнях.

Події є пасивними елементами в EPC.

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

Функції є активними елементами в EPC.

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

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

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

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

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

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

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

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

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

Розглянемо алгоритм побудови діаграми:

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

Функція



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

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

Подія



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

Стрілка



відображає зв'язки елементів діаграми EPC процесу між собою.

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

Суб'єкт



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

Паперовий документ



використовується для відображення на діаграмі паперових документів, що супроводжують

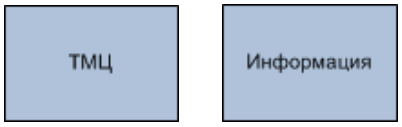
виконання функції. Усередині блоку міститься найменування паперового документа.

Електронний документ



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

ТМЦ / Інформація



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

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

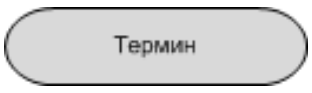


База даних



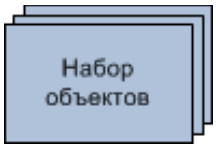
використовується для відображення діаграми бази даних, що супроводжує виконання функції. Усередині блоку міститься найменування бази даних.

Термін / Статус об'єкта

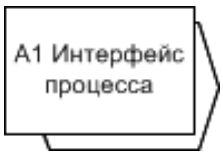


елемент може бути використаний для позначення даних, що передаються між процесами або оброблюваних при виконанні процесів. Елемент може бути також використаний для позначення статусів паперових/електронних документів та інших елементів

Набір об'єктів



Інтерфейс процесу



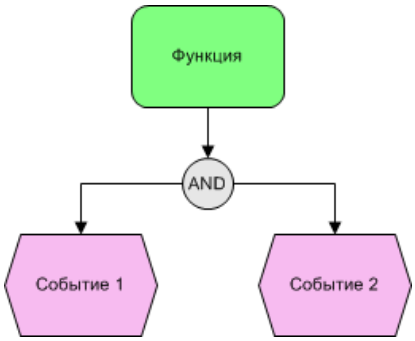
елемент, що позначає зовнішній (стосовно поточної діаграми) процес або функцію. Усередині блоку міститься найменування зовнішнього процесу.

Використовується для зазначення взаємозв'язку процесів:

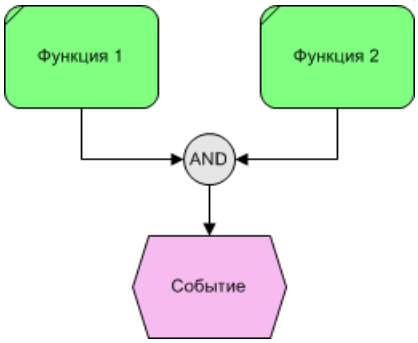
– позначає попередній або наступний процес по відношенню до діаграми процесу;
– позначає процес, звідки надійшов або куди передається об'єкт.

Оператор "І" 

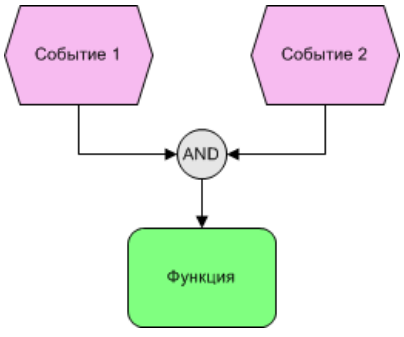
використовується для позначення злиття/розгалуження як функцій, так і подій. Якщо завершення виконання функції має ініціювати одночасно кілька подій, це позначається за допомогою оператора "І", наступного після функції і перед подіями.



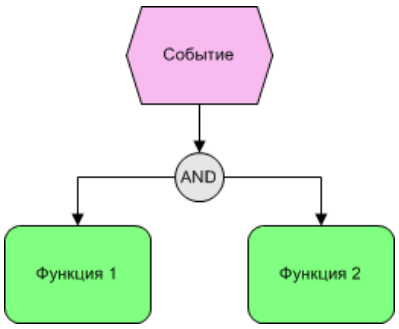
Якщо подія відбувається лише після обов'язкового завершення виконання кількох функцій, це позначається за допомогою оператора "І", наступного після функцій і перед поодинокою подією.



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



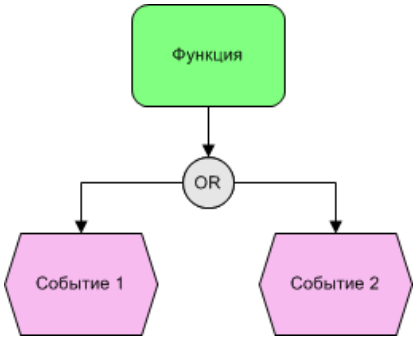
Якщо одна подія може ініціювати одночасне виконання кількох функцій, це позначається за допомогою оператора "І", наступного після події і перед функціями.



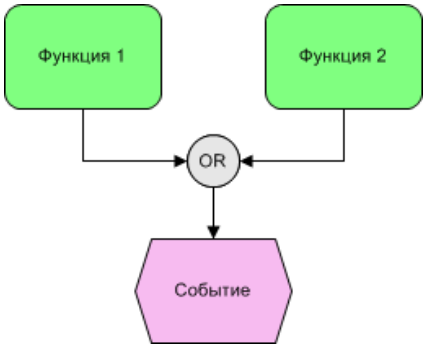
Оператор "АБО" 

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

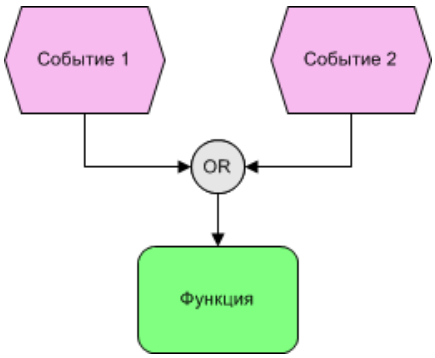
Якщо завершення виконання функції може ініціювати одну або кілька подій, це позначається за допомогою оператора "АБО", наступного після функції та перед подіями.



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



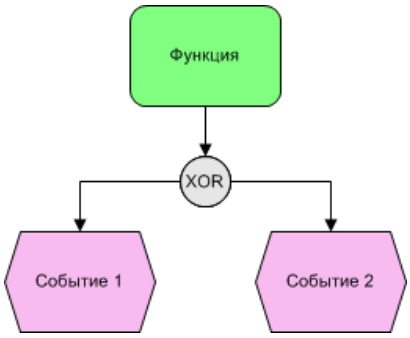
Якщо функція може почати виконуватися після того, як відбудеться одна або кілька подій, це позначається за допомогою оператора "АБО", наступного після подій і перед функцією.



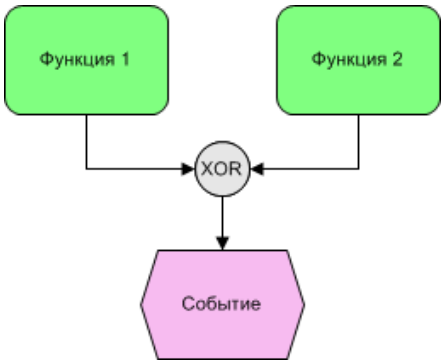
Оператор "Виключне АБО" 

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

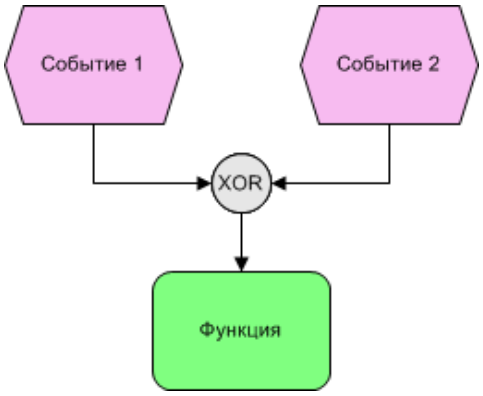
Якщо завершення виконання функції може ініціювати лише одну з подій залежно від умови, це позначається за допомогою оператора "Виключне АБО", наступного за функцією і перед подіями.



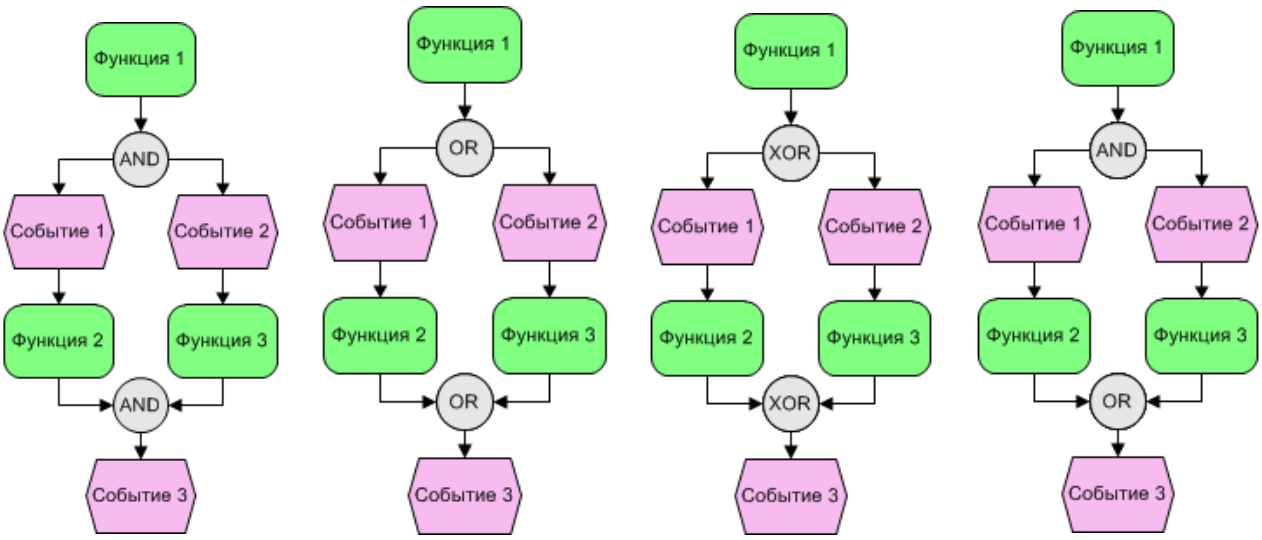
Якщо подія відбувається відразу після завершення виконання однієї функції, або іншої, то це позначається за допомогою оператора "Виключне АБО", наступного після функцій і перед одиночною подією.



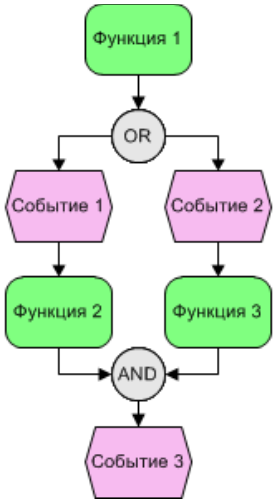
Якщо функція може почати виконуватися відразу після того, як відбудеться або одна подія, або інша, то це позначається за допомогою оператора "Виключне АБО", наступного після кількох подій та перед функцією.



Приклади допустимих ситуацій:



Приклад неприпустимої ситуації:

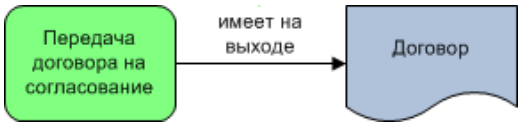


Типи зв'язків, які можна навести між елементами на діаграмі EPC:

– змінює



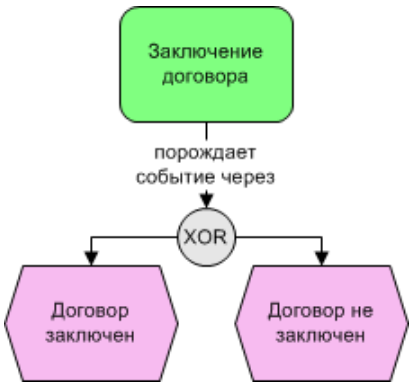
– має на виході



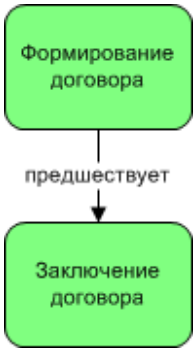
– створює на виході



– має, породжує подію через xor



– передує



– породжує



– розподіляє



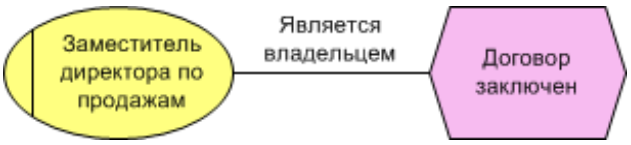
– зчитує



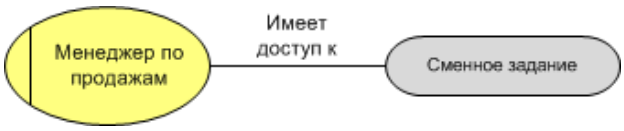
– виконує



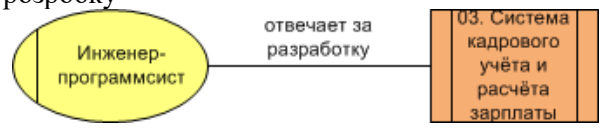
– є власником



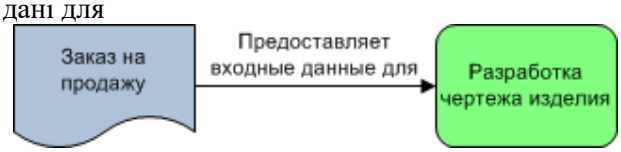
– має доступ до



– відповідає за розробку



– надає вхідні дані для



– встановлює статус



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

– Переважно розміщення схеми зверху донизу. Допустимо - зліва направо.
– Елементи виконавців маємо праворуч від функцій.
– Ліворуч та вгору від функції даємо інформацію про вхідні документи.
– Вліво та вниз від функції представляємо інформацію про вихідні документи.
– Вниз і вправо від функції маємо блок «Інформація».
– Вгору та вправо від функції представляємо дані про програми.
– Довільне розташування переважно для елементів «База даних» та «Картотека».
– Дані про матеріальні потоки подаємо зліва про відомості про документи, що їх супроводжують.

2. Завдання до лабораторної роботи 10

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. Моделювання бізнес-процесів. URL: https://stud.com.ua/87161/ekonomika/modelyuvannya_biznes-protsesiv
3. Методологія структурного аналізу та проектування SADT. URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?id=39227
4. Нотация IDEF0. URL: http://www.businessstudio.com.ua/bp/bs/overview/notation_idef0.php
5. Business Process Modeling Notation (BPMN). Version 1.2. – Object Management Group, 2009. – 316 p.
6. Brackett, J. Applying SADT to Large System Problems / J.Brackett, C. McGowan. – SofTech Technical Paper TP059,January 1977.
7. Постіл С. Д. UML. Уніфікована мова моделювання інформаційних систем: навчальний посібник. - Ірпінь : Ун-т держ. фіск. служби України, 2019. - 321 с.
8. J. Jürjens. Secure systems development with UML. – Berlin: Springer; Heidelberg, 2005. – 309 p.