МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

О. К. БІЛОШИЦЬКА

ПРОЄКТУВАННЯ МЕДИЧНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

ПРАКТИЧНІ РОБОТИ

*Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів першого (бакалаврського) рівня вищої
освіти за освітньою програмою «Медична інженерія»
спеціальності 163 «Біомедична інженерія»*

Київ
КПІ ім. Ігоря Сікорського
2021

Рецензенти *Зюков О. Л.*, доктор медичних наук, професор, головний лікар, заступник директора з клінічної роботи, провідний науковий співробітник наукового відділу організації медичної допомоги, Державна наукова установа «Науково-практичний центр профілактичної та клінічної медицини» Державного управління справами
Носовець О. К., кандидат технічних наук, доцент кафедри біомедичної кібернетики, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
Відповідальний редактор *Соломін А. В.*, к.ф.-м.н., доцент, доцент кафедри біомедичної інженерії КПІ ім. Ігоря Сікорського

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 1 від 16.09.2021 р.)
за поданням Вченої ради факультету біомедичної інженері
(протокол №16 від 30.08.2021 р.)

Електронне мережне навчальне видання

Білошицька Оксана Костянтинівна, канд. техн. наук

ПРОЄКТУВАННЯ МЕДИЧНИХ ІНФОРМАЦІЙНИХ СИСТЕМ
ПРАКТИЧНІ РОБОТИ

Проєктування медичних інформаційних систем: Практичні роботи [Електронний ресурс] : навч. посіб. для студ. спеціальності 163 «Біомедична інженерія» / О. К. Білошицька; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 4,6 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2021. – 84 с.

Навчальний посібник призначено для закріплення на практичних заняттях теоретичних знань у проєктуванні медичних інформаційних систем, набуття практичних навичок щодо розробки блок-схем в різних нотаціях моделювання відповідно до затверджених стандартів для подальшого проєктування та розроблення медичних інформаційних систем. Виконання практичних робіт здійснюється на основі затверджених МОЗ форм медичних документів.

© О. К. Білошицька, 2021
© КПІ ім. Ігоря Сікорського, 2021

Зміст

Вступ

Основною метою навчальної дисципліни «Проектування медичних інформаційних систем» є формування у студентів здатності розробляти алгоритми моделювання бізнес-процесів в медичних інформаційних системах; застосовувати методи і алгоритми вирішення теоретичних і прикладних задач в області реалізації медичних інформаційних систем; розробляти комплекси формалізації та управління медичною інформацією.

Отримані знання в результаті вивчення дисципліни можуть бути використані для спрощення роботи працівників закладу охорони здоров'я за рахунок розробки медичних інформаційних систем та реалізації бізнес-процесів в їх діяльності.

Дисципліна «Проектування медичних інформаційних систем» своїм змістом об'єднує, інтегрує, синтезує, узагальнює та закріплює знання, які були отримані студентами при вивченні інших загальних та спеціальних фахових дисциплін.

При вивченні даної дисципліни раніше відомі факти розглядаються детальніше та на більш високому науковому рівні. В методичних вказівках до всіх практичних робіт спочатку наводиться теоретичний виклад теми роботи. Це спрощує розуміння та виконання прикладної (практичної) частини роботи.

Порядок виконання практичної роботи полягає в наступному:

1. Перед початком виконання роботи студенти вивчають опис практичної роботи.

2. Після ознайомлення з теоретичним матеріалом студенти виконують завдання практичної роботи.

3. Практична робота вважається виконаною, якщо вона захищена. При захисті практичної роботи викладач може спитати не лише про суть виконання роботи та її результатах, але і теоретичний матеріал того розділу, до якого відноситься дана робота.

4. Всі закріплені знання, які отримані під час виконання практичних робіт, студенти показують при виконанні індивідуального завдання, яке передбачено силабусом (робочою програмою) дисципліни.

Варіанти для виконання практичних робіт (визначається викладачем):

1. Виписка з МКАХ
2. Декларація
3. Довідка для одержання санаторно-курортної путівки
4. Довідка про тимчасову непрацездатність учня або студента
5. Екстрене повідомлення
6. Індивідуальна карта вагітної і породіллі
7. Історія пологів
8. Історія розвитку дитини
9. Карта звернення за антирабічною допомогою
10. Карта обліку диспансеризації
11. Карта обстеження дитини (підлітка) з незвичайною реакцією на вакцинацію (ревакцинацію) БЦЖ
12. Карта профілактичних щеплень
13. Карта хворого у фізіотерапевтичному кабінеті
14. Карта хворого, який вибув з стаціонару
15. Карта хворого, який підлягає лікуванню променевої терапії
16. Консультативний висновок
17. Контрольна карта диспансерного хворого
18. Листок непрацездатності
19. Лікарське свідоцтво про смерть
20. Лікарсько-контрольна карта фізкультурника і спортсмена
21. Лікувальна карта призовника і відрізний талон лікувальної карти призовника
22. Медична довідка від'їжджаючого за кордон
23. Медична довідка для учня або студента

24. Медична довідка на дитину у заклад

25. Медична карта новонародженого

26. Медична карта огляду осіб для визначення спроможності займатися відповідним видом діяльності за станом здоров’я

27. Медична карта профілактичного наркологічного огляду

28. Медична карта стаціонарного хворого

29. Медична карта стоматологічного хворого

30. Направлення на МСЕК

31. Обмінна карта вагітної

32. Первинний огляд анестезіолога і протокол загального знеболювання

33. Протокол (карта) патологоанатомічного дослідження

34. Реєстраційна генетична карта

35. Реєстраційна карта вагітної, яка хворіє на цукровий діабет

36. Реєстраційна карта хворого з хронічною хворобою нирок (ХХН) або трансплантованою ниркою

37. Санаторно-курортна карта для дітей і підлітків

38. Санаторно-курортна картка

Практична робота №1
Створення моделі медичного документу у нотації IDEF0

Мета роботи – оволодіння прийомами функціонального моделювання заданої предметної області, отримання навичок створення і редагування функціональних моделей у програмі Edraw Max.

Теоретичні відомості

Стислі відомості щодо методології IDEF0

Склад та основні елементи IDEF0

Модель у нотації IDEF0 – це сукупність ієрархічно упорядкованих та взаємопов'язаних діаграм. Кожна діаграма є одиницею опису системи і розташовується на окремому аркуші. Сама модель IDEF0 включає чотири основних елементи, а саме:

- функціональний блок;
- інтерфейсна дуга (стрілка);
- декомпозиція;
- глосарій.

Функціональний блок IDEF0

Функціональні блоки позначають зазначені процеси, функції або завдання, що відбуваються протягом певного часу і мають реальні результати.

Функціональні блоки зображують у вигляді прямокутників. Кожен блок має ім'я. Імена блоків мають бути унікальні. В назві блоку поєднують дієслівний іменник, що позначає процес, або саме дієслово. Приклади блоків у нотації IDEF0 показані на рисунку 1.1.

Рисунок 1 – Приклади блоків з назвами робіт

Кожний функціональний блок моделі має унікальний номер. Цей номер складається з префіксу й числа. Префікс може бути довільної довжини. На практиці за префікс часто беруть символ «А». В цьому випадку контекстна (коренева) робота моделі (тобто її функціональний блок) позначають «А0». Визначення функціональних блоків заносять до глосарію або до словника робіт (т. з. Activity Dictionary).

Інтерфейсна дуга (стрілка або Arrow)

Взаємодію функціональних блоків (між собою або з елементами зовнішнього середовища) відображають у вигляді інтерфейсних дуг (або стрілок). Кожна дуга має своє позначення у вигляді іменника. Наприклад, – «Заготівля», «Продукт», «Управління» і т. п. Для позначення стрілок в IDEF0 можна використовувати іменовані поєднання. Наприклад, – «Готовий продукт», «Активний процес», «Дефіцитний ресурс» і т. д. Назви дуг заносять до спеціального словника, – глосарію стрілок (Arrow Dictionary).

У моделі IDEF0 є 4 різновиди стрілок. Кожний різновид стрілки має своє розташування навколо функціонального блоку (рис. 1.2).

Вхід (Input), – ресурс (матеріальний, нематеріальний, інформаційний) що буде перетворений процесом до кінцевого результату (вихід). За своєю сутністю «Вхід» відповідає на питання: «Що буде обробляти процес?». Стрілка Input входить до лівої грані процесу (роботи).

Управління (Control), – означає правила, стратегії, процедури або стандарти, якими керується процес (робота). Стрілка «Управління»

відповідає на питання: «Що буде керувати виконанням процесу?». Кожна робота мусить мати хоча б одну стрілку управління. Стрілка «Управління» входить у верхньої грані процесу (роботи).

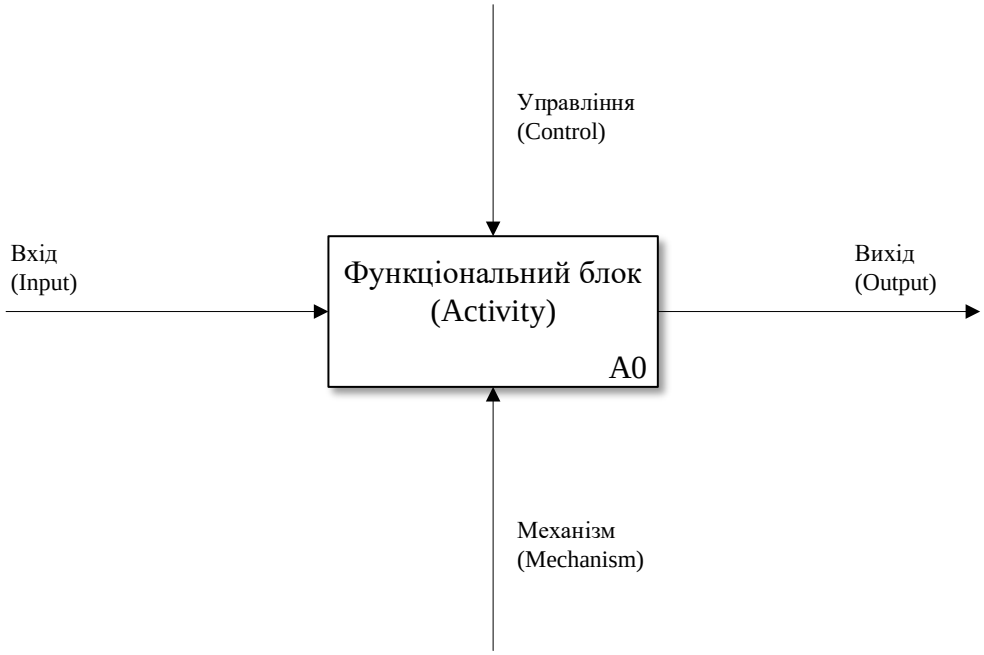


Рисунок 1.2 – Різновиди стрілок у нотації IDEF0

Вихід (Output), – результат виконання процесу. Результат може бути матеріальним або нематеріальним (наприклад, – інформаційним). Стрілка «Вихід» показує: «Що ми отримаємо після виконання процесу?». Кожна робота мусить мати хоча б одну стрілку виходу (процес без результату не має сенсу). Стрілку «Вихід» ставлять як вихідну із правої грані роботи.

Механізм (Mechanism), – це ресурси (матеріальні, нематеріальні), які використовує процес для отримання результату. Це можуть бути трудові ресурси, інформація, обладнання тощо. Стрілка «Механізм» відповідає на питання: «За рахунок чого буде виконаний процес і отримано кінцевий результат?». Стрілку зображують як вхідну до нижньої межі роботи.

Глосарій

Для кожного з елементів IDEF0 (діаграм, функціональних блоків, інтерфейсних дуг) мають бути створенні та і підтримуватись відповідні

набори визначень, ключових слів, коментарів тощо. Ці визначення називають глосарієм. Їх завдання – детально охарактеризувати об’єкт, що відображений конкретним елементом, пояснити сутність цього елементу. Наприклад, для інтерфейсної дуги «Розпорядження про оплату» глосарій може містити перелік полів відповідного документу, набір віз, дозволів тощо. Глосарій гармонійно доповнює графічну візуалізацію моделі, додає необхідну інформацію та робить модель простішою для сприйняття.

Декомпозиція

Ця процедура має за мету розбиття системи на окремі фрагменти (функції, підфункції, тощо) до рівня конкретних процедур. Таким чином, модель IDEF0 представляє собою серію діаграм із супровідною документацією, що розбивають складний об’єкт на складові частини і показують їх у вигляді блоків. Деталі кожного з основних блоків відображені на інших діаграмах. Кожна детальна діаграма є декомпозицією блоку з діаграми вищого рівня. Діаграма вищого рівня, що підлягає декомпозиції, називається для одного або декількох деталізованих процесів.

Вхідні та вихідні дуги на діаграмі верхнього рівня мають повторювати дуги для діаграми нижнього рівня (оскільки обидві вони описують одну й ту саму частину системи).

Різновиди діаграм IDEF0

Модель IDEF0 може включати 4 типи діаграм:

- **контекстну** (в кожній моделі може бути тільки одна контекстна діаграма);
- **декомпозиції**;
- **дерева вузлів**;
- **експозиції** (FEO).

Контекстна діаграма є вершиною деревовидної структури діаграм і представляє собою загальний опис системи та її взаємодії із зовнішнім середовищем.

Після опису системи в цілому проводиться розбиття її на окремі фрагменти. Цей процес називається функціональною декомпозицією, а діаграми, що описують кожен фрагмент і взаємодію фрагментів, – **діаграмами декомпозиції**. Після декомпозиції контекстної діаграми проводиться декомпозиція кожного великого фрагмента системи на більш детальні і т. д. аж до потрібного рівня описання кожного процесу.

Діаграма дерева вузлів відображає ієрархію робіт (їх підпорядкованість), але не показує взаємозв'зки між роботами.

Діаграми для експозиції (FEO) потрібні для ілюстрації окремих фрагментів моделі. Вони дозволяють показати альтернативну точку зору або ж слугують для спеціальних цілей. Усі діаграми мають бути пронумеровані. Контекстна діаграма має номер А-0, декомпозиція контекстної діаграми – номер А, інші діаграми декомпозиції – номери відповідно до вузлу (А1, А2, А21, тощо).

Реалізація функціональної моделі IDEF0 у програмі EdrawMax 9.2

Для побудови функціональної моделі бізнес-процесу в EdrawMax нам знадобляться такі блоки.

Блок заголовку – рамка, що займає весь аркуш, її треба оформити відповідно до правил формування діаграм в нотації IDEF0.

Блок тексту, – потрібен для описання точки зору і мети на контекстній діаграмі.

Блок дії – слугує для описання робіт у межах одного процесу.

Одностороннє з'єднання – показує інтерфейсні дуги (вхід, вихід, механізм, управління).

Лінія з'єднання IDEF0 – використовується для зображення інтерфейсних дуг між роботами у моделі.

Щоб відкрити доступ до вказаних блоків в програмі EdrawMax робимо так.

1. Через меню «Пуск» викликаємо EdrawMax 9.2 («Пуск» – «EdrawMax»). Відкриється головне вікно програми.

2. У цьому вікні вибираємо категорію «Flowchart» обираємо шаблон «IDEF Diagram», натискаємо «Create» (рис. 1.3).

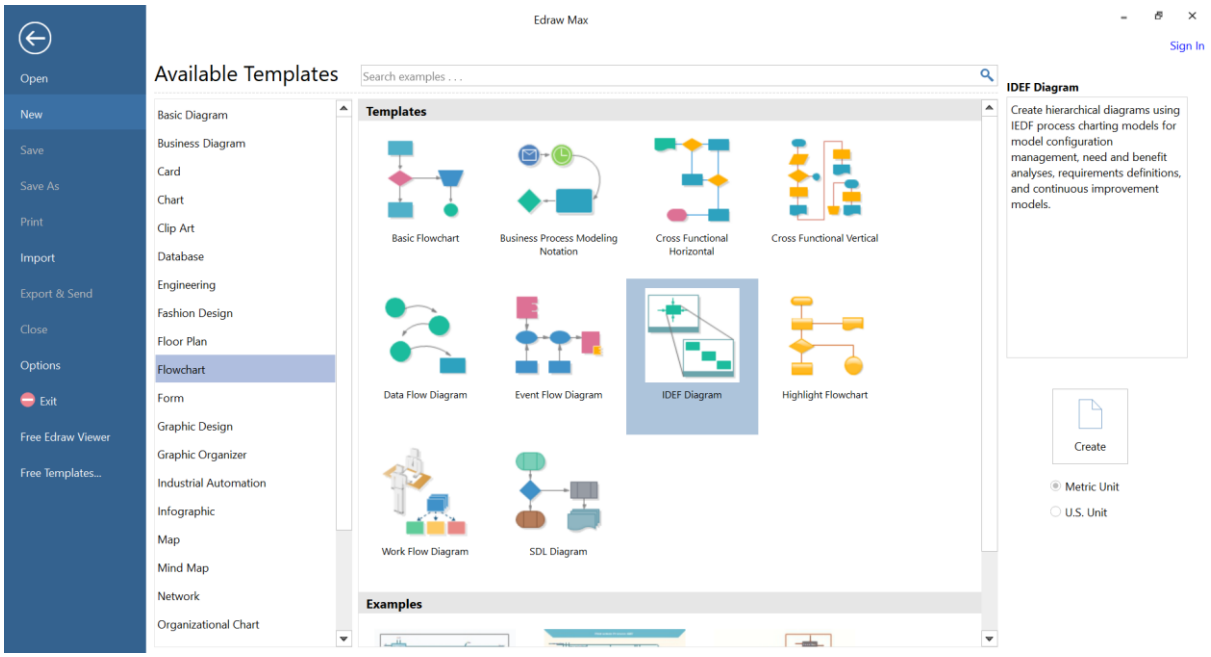


Рисунок 1.3 – Активізація елементів EdrawMax

Зліва у меню будуть доступні основні елементи IDEF0 для використання.

Методика виконання роботи

Розглянемо розробку функціональної моделі для реалізації бізнес-плану лікувально-профілактичного закладу (ЛПЗ).

Наше завдання: створити функціональну цього модель процесу. Зібрати необхідну інформацію з заданої форми медичної документації, ознайомитися з інструкцією по заповненню форми, при необхідності скоригувати, визначити показники зведеного плану ресурсів, скласти плани залучення і розміщення ресурсів як по ЛПЗ в цілому, так і по його відділеннях.

Порядок виконання роботи

Загальний порядок розробки функціональної моделі має два етапи і виглядає так:

1) виділення функціональних блоків (функцій процесу);
2) виділення зв'язків між функціями.

Побудову функціональної моделі починаємо зі створення контекстної діаграми, що буде описувати глобальну функцію: розробка плану формування облікової форми медичної документації та її зв'язки із зовнішнім світом.

Створення контекстної діаграми

Для будування контекстної діаграми робимо так.

1. Через меню «Пуск» викликаємо програму EdrawMax.

2. Вибираємо «File» – «New» – «Flowchart» – «IDEF Diagram», натискаємо кнопку «Create». Вікно програми набуде вигляду, як на рисунку 1.4.

3. За умовчанням буде ввімкнена альбомна орієнтація. Якщо ввімкнена книжна – змініть орієнтацію на альбомну. Для цього викликаємо меню «Page Layout», клацаємо на кнопку «Orientation», із запропонованого списку вибираємо «Landscape».

4. Утримуючи ліву кнопку миші, перетягуємо елемент «Title block» («Блок заголовка») на порожню сторінку (рис. 1.5).

5. У вікні заповнюємо дані фігури: вводимо номер контекстної діаграми, а також ім’я процесу (в нашому випадку це: «А», «Форма медичної документації ф.025/о»). Тепер ім’я заголовка фігури «Блок заголовка» має відповідати номеру і назві завдання, декомпозиція якого буде зображена в даній області. Наприклад: «А1», «Складові МКАХ».

6. За допомогою елементів 1 legged connector («Одностороннє з’єднання») додаємо стрілки на контекстну діаграму відповідно до таблиці 1.1. Результат наших дій показано на рисунку 6.

Таблиця 1.1 – Стрілки контекстної діаграми

Ім’я стрілки (Arrow Name)	**Тип стрілки** (Arrow Type)
Паспортні дані пацієнта	Вхід (Input)
Анамнестичні дані пацієнта	Вхід (Input)
Дані попередніх досліджень пацієнта	Вхід (Input)
Закони України	Управління (Control)
Затверджена інструкція по заповненню форми	Управління (Control)
Стандарти надання медичної допомоги	Управління (Control)
Лікарі	Механізм (Mechanism)
Медичні сестри	Механізм (Mechanism)
Реєстратори	Механізм (Mechanism)
Призначення, рекомендації, лікування	Вихід (Output)

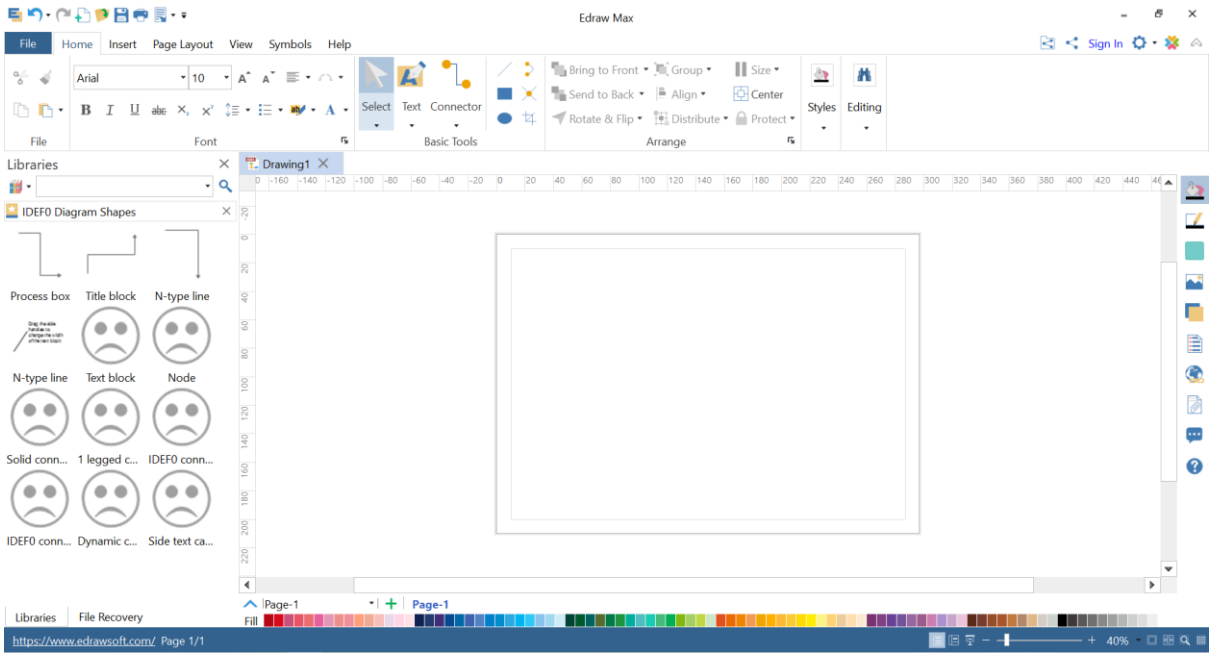


Рисунок 1.4 – Головне вікно програми

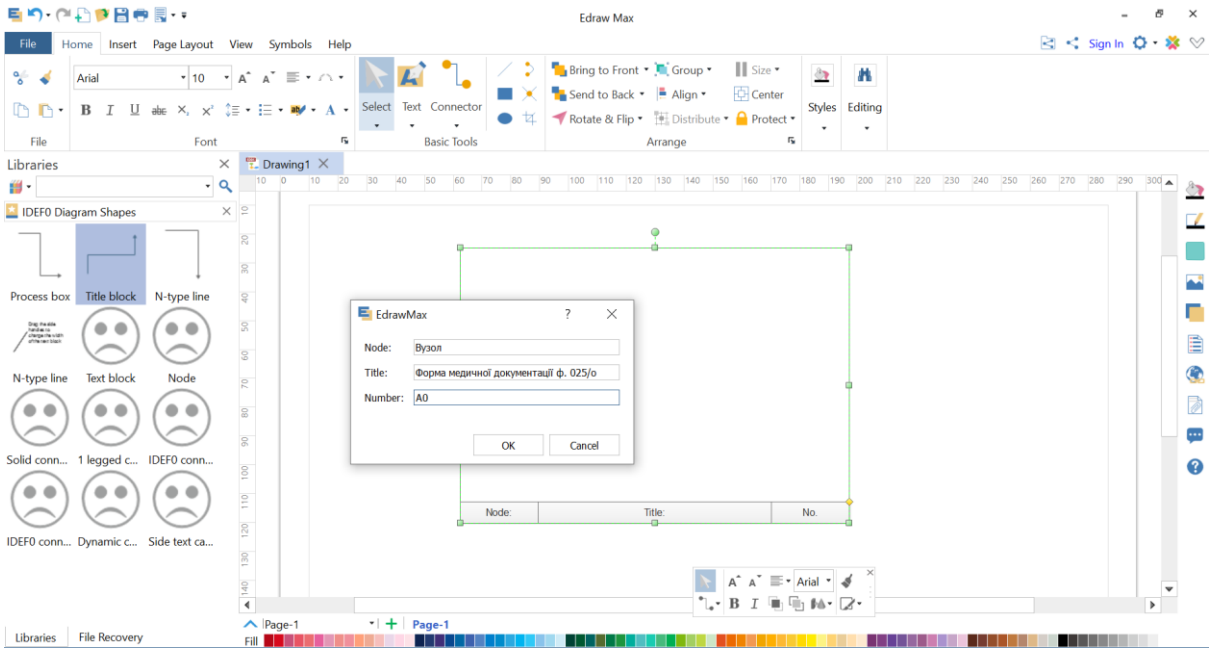


Рисунок 1.5 – Додавання елементу «Блок заголовка»

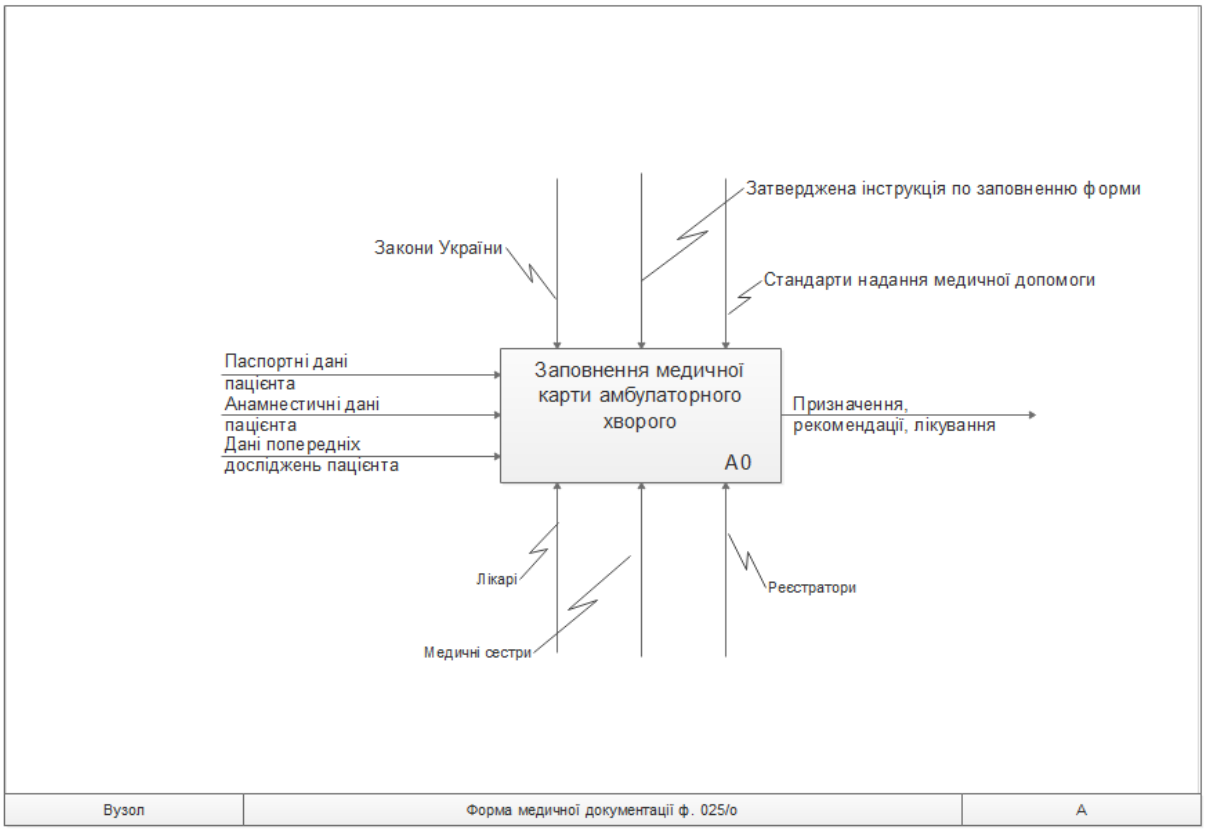


Рисунок 1.6 – Контекстна діаграма

Далі робимо декомпозицію функції *«Заповнення медичної карти амбулаторної хворого»* на ряд складових, кожна з яких описує певний процес.

Створення діаграми декомпозиції

Щоб сформувати діаграму декомпозиції в Edraw Max робимо так.

1. Клацаємо на значку «+» в лівій нижній частині рядка стану або натискаємо на вкладку Insert, далі обираємо пункт Blank Page та натискаємо Blank Page, або можна скористатися комбінацією клавіш Alt+Shift+N. У документі Edraw Max з’явиться нова сторінка. За замовчуванням вона буде називатися «Page-X», де X її номер у документі.

2. Клацаємо на імені сторінки правою кнопкою миші, з контекстного меню вибираємо Rename Page («Перейменувати сторінку»).

3. Вводимо назву сторінки відповідно до рівня декомпозиції, наприклад «А0», «А1», тощо.

4. Виконуємо декомпозицію А0, тобто ділимо роботи на діаграми в області «Блок заголовка». Рівень функціональної моделі у нашому випадку буде складатися з 8 основних блоків (розділів МКАХ), а саме:

- Сигнальні позначки;
- Листок запису заключних (уточнених) діагнозів;
- Відомості про щеплення;
- Листок профілактичного огляду;
- Строки тимчасової непрацездатності;
- Інформація про госпіталізацію;
- Відомості щодо страхування;
- Щоденник.

Для детального відображення покажено взаємодію 4 блоків (Сигнальні позначки, Листок запису заключних (уточнених) діагнозів; Інформація про госпіталізацію; Щоденник) та титульної сторінки МКАХ.

Розділяємо стрілки для діаграми декомпозиції відповідно до контекстної діаграми. Для цього копіюємо вхідні і вихідні стрілки, що пов’язані з роботою вищого рівня до поля декомпозиції.

Розділення стрілок. Для розгалуження стрілки потрібно провести (намалювати) нову стрілку, що складається з декількох блоків за принципом «Односторонній зв’язок». Напрямок малювання – від фрагмента стрілки до сегмента роботи.

Наприклад, елемент «Закон України» потрібен, щоб всі розділи медичної карти амбулаторного хворого заповнювалися відповідно до законодавства України. В цьому випадку проводимо однойменну стрілку до всіх робіт.

Злиття стрілок. Для злиття двох стрілок треба провести роботи, що аналогічні розгалуженню.

Розстановка ICOM-міток. Для реалізації цієї дії використовуйте блоки тексту.

Результат виконання вказаних пунктів показано на рисунку 1.7.

Зробимо декомпозицію наступного блоку функціональної моделі. Він називається «Заповнення щоденника огляду пацієнта при зверненні з лікувально-діагностичною метою». Новий рівень декомпозиції буде складатися з чотирьох функціональних блоків (рис. 1.8):

– Заповнення даних об’єктивного обстеження;
– Встановлення попереднього/заключного діагнозу;
– Складання плану обстеження;
– Складання плану лікування.

Результат декомпозиції моделі показано на рисунку 1.8.

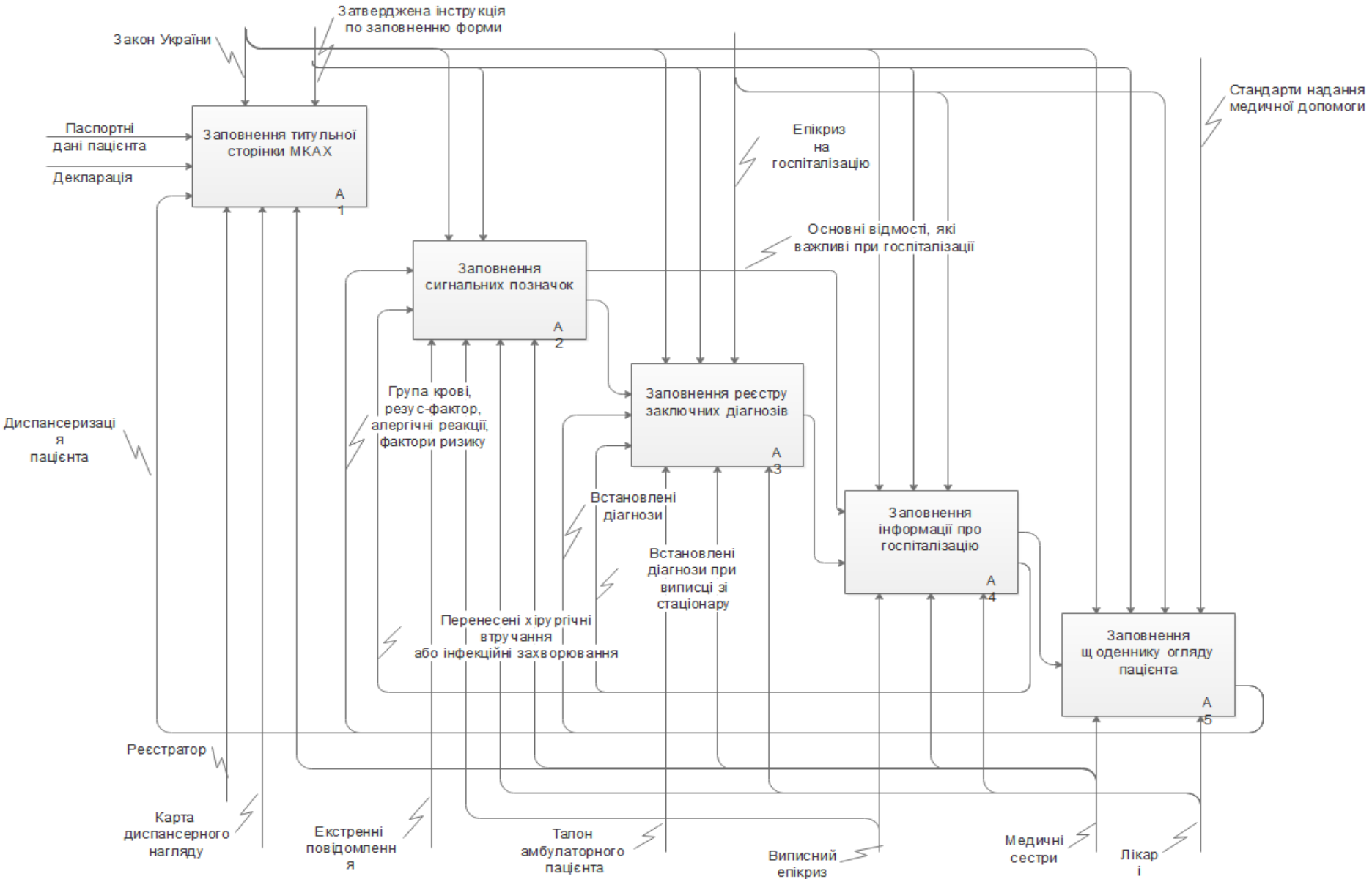


Рисунок 1.7 – Діаграма декомпозиції блоку A0 (декомпозиція 1-го рівня)

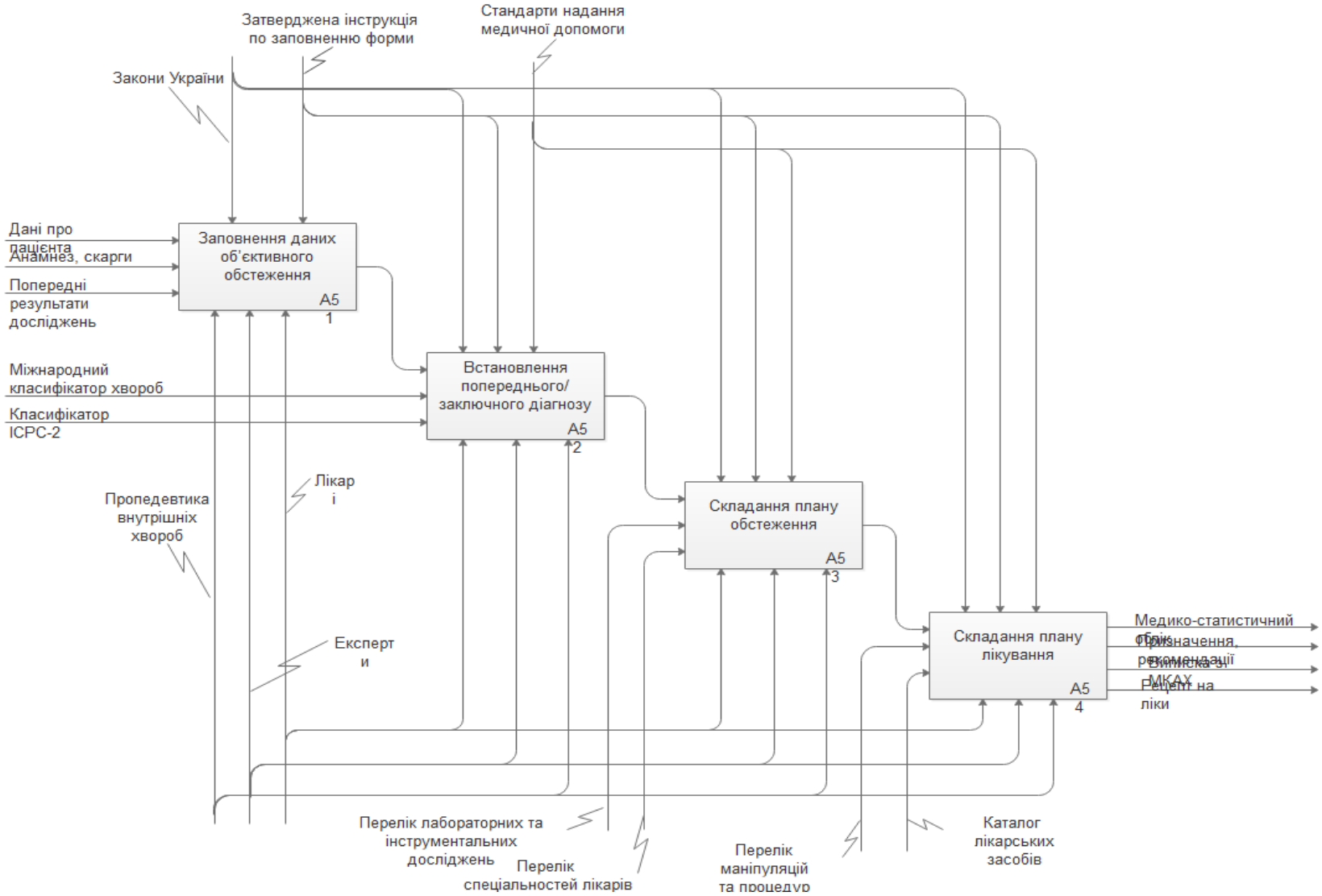


Рисунок 1.8 – Другий рівень функціональної моделі (декомпозиція 2-го рівня)

Індивідуальне завдання

Засобами Edraw Max розробити функціональну модель у нотації IDEF0 для свого варіанту (відповідної медичної форми облікової документації):

1. Побудувати контекстну діаграму згідно прикладу (рис. 1.6).
2. Побудувати 2 діаграми декомпозиції верхнього процесу (рис. 1.7-1.8).

Для побудови функціональних моделей медичних форм облікової документації використовувати затверджену форму медичної документації та інструкцію до її заповнення. З інструкції потрібно дізнатися, які елементи будуть входами, управлінням, механізмами та виходом.

Наприклад, якщо при заповненні медичної карти амбулаторного хворого (форма 025/0) на титульній сторінці використовуються паспортні дані пацієнта, то вони виступатимуть входом.

Основними елементами управління для всіх форм медичної документації будуть як мінімум Закони України (згідно яких затверджено форму документації), Інструкції по заповненню відповідної форми (щоб лікарі та користувачі знали, яку інформацію заносити у відповідні поля), стандарти надання медичної допомоги (лікарі повинні знати, як лікувати те чи інше захворювання, що призначати, як діагностувати і т.д.).

Основними механізмами будуть виступати медичні працівники (лікарі, медичні сестри, реєстратори і т.д.), які будь-яким чином впливають на процес формування медичної документації.

Виходами будуть виступати ті елементи, які можуть бути отримані після виконання основних процедур з медичною документацією. Наприклад, після заповнення щоденника огляду пацієнта, який звернувся вперше в році для проходження профогляду, лікарем як мінімум буде складено план обстеження (консультації лікарів, направлення на дослідження і т.д.), видано рекомендації, призначено лікування, видано довідки/виписки і т.д.

Практична робота №2
Моделювання потоків даних

Мета роботи – оволодіння прийомами моделювання потоків даних для медичних облікових форм у програмі Edraw Max.

Теоретичні відомості

Основним засобом моделювання функціональних вимог до системи, яка проєктується, є діаграми потоків даних (DFD). За їх допомогою ці вимоги представляються у вигляді ієрархії функціональних компонентів (процесів), пов’язаних потоками даних.

Діаграма потоків даних – це графічний інструмент, який дозволяє системним аналітикам відобразити потік даних в інформаційній системі. Це наочна картина, яка показує звідки і куди йдуть дані.

Головна ціль такого представлення – продемонструвати, як кожний процес перетворює вхідні дані в вихідні, а також виявнити відношення між цими процесами.

Модель DFD, як і більшість інших структурних моделей – ієрархічна модель. Кожний процес може бути підданий декомпозиції, тобто розбиттю на структурні складові, відношення між якими в тій же нотації можуть бути показані на окремій діаграмі.

Для побудови DFD використовуються дві різні нотації, які незначно відрізняються один від одної графічним зображенням символів і відповідають методам Йордона-Де Марко та Гейна-Серсона.

В основі методології DFD лежить побудова моделі, яка реально існує, або ІС, яка проєктується. Відповідно до методології модель системи визначається, як ієрархія діаграм потоків даних, які описують асинхронний процес перетворення інформації від її введення в систему до видачі користувачу. Діаграми верхніх рівнів ієрархії називаються контекстними діаграмами, які визначають основні процеси або підсистеми ІС з зовнішніми

входами і виходами. Ці діаграми можуть деталізуватися за допомогою діаграм нижнього рівня до тих пір, поки не буде досягнуто такий рівень декомпозиції, на якому процеси стають елементарними і деталізувати їх далі неможливо.

Зовнішні сутності представляють собою матеріальний предмет або фізична особа, наприклад, замовники, медичний персонал, пацієнти і є джерелами або споживачами інформації. Як джерела інформації, вони породжують інформаційні потоки (потоки даних), які переносять інформацію до підсистем або процесам. Процеси (підсистеми) перетворюють інформацію та породжують нові потоки, які переносять інформацію до інших процесів або підсистемам, накопичувачам даних або зовнішнім сутностям – споживачам інформації.

Таким чином, основними компонентами діаграми потоків даних є:

- зовнішні сутності;
- системи/підсистеми;
- процеси;
- накопичувачі даних;
- потоки даних.

Визначення деякого об’єкту або системи в якості зовнішньої сутності вказує на те, що вона знаходиться за межами ІС, яка аналізується.

При побудові моделі складної ІС вона може бути представлена в загальному виді на так званій контекстній діаграмі у вигляді однієї системи як єдиного цілого, або може бути декомпозована на ряд підсистем.

Процес представляє собою перетворення вхідних потоків даних у вихідні відповідно до певного алгоритму. Фізично процес може бути реалізовано різними способами: це може бути відділення закладу охорони здоров’я (відділ), який виконує оброблення вхідних документів та випуск звітів, програма, апаратно реалізований логічний пристрій і т.д. Інформація в полі фізичної реалізації показує, яке відділення, програма або апаратний пристрій виконує даний процес.

Накопичувач даних представляє собою абстрактний пристрій для збереження інформації, яку можна в будь-який момент помістити в накопичувач і через деякий час вилучити. Накопичувач даних є праобразом майбутньої бази даних, та опис даних, який зберігається в ньому, повинні бути скоординовані з інформаційною моделлю.

Потік даних визначає інформацію, яка передається шляхом деякого з'єднання від джерела до приймача.

Хто використовує діаграми потоків даних?

1. Системні аналітики.
2. Кінцеві користувачі.
3. Розробники баз даних.
4. Системні програмісти.

Як виглядає діаграма потоків даних?

Схема потоку даних складається з основних геометричних фігур, які з'єднані стрілками, що показують напрямок потоку (рис. 2.1).

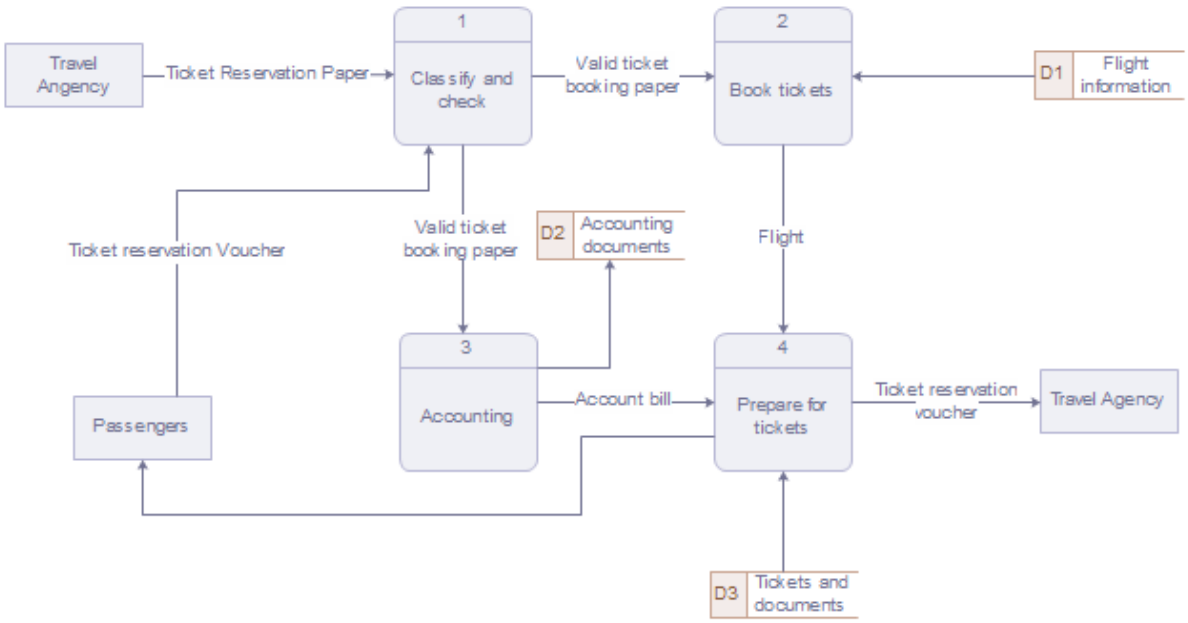
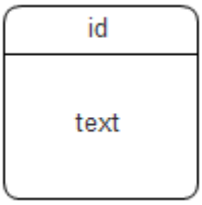


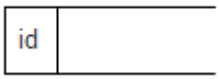
Рисунок 2.1 – Загальний вигляд діаграми потоків даних

Основні символи схеми потоків даних

Процес – процес або задача, яка виконується системою, наприклад, створення, зміна, збереження або видалення.

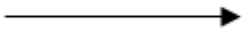


Сховище даних – місце, де зберігаються дані. Може бути файлом або базою даних.



Сутність – це джерело або місце призначення даних. Сутності є зовнішніми по відношенню до системи.

Data Flow – це потік даних.

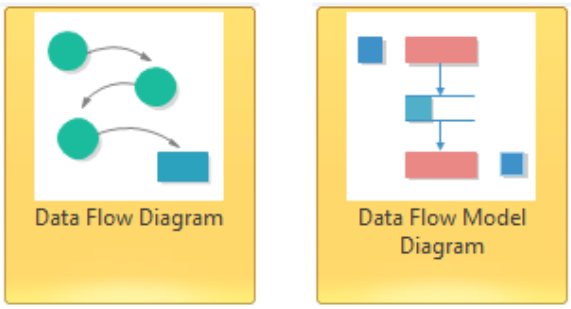


Принципи побудови діаграми потоків даних

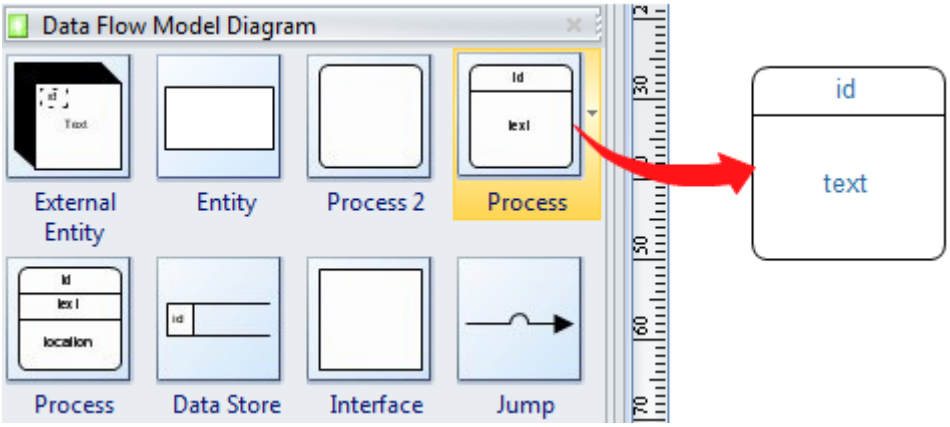
1. **Повнота:** відноситься до степені, в якій всі необхідні компоненти діаграми потоку даних були включені та повністю описані.
2. **Узгоджуваність:** відноситься до степені, в якій інформація, яка міститься на одному рівні набору вкладених діаграм потоків даних, також включається на інші рівні.

Хід виконання роботи

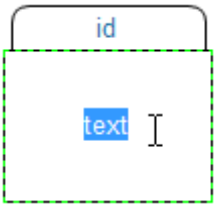
1. Запустіть програму Edraw Max. В категорії «Програмне забезпечення» («Software») виберіть «Діаграма потоку даних» (Data Flow Model Diagram).



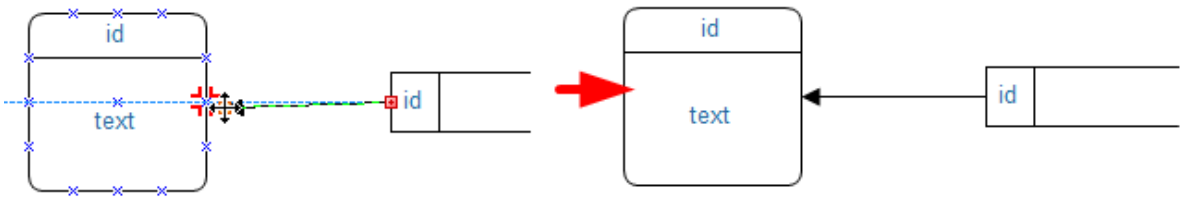
2. З бібліотеки символів потоку даних (автоматично відкривається при запуску нової сторінки документа) перетягніть необхідні символи на сторінку документу.



3. Додайте текст (двічі натисніть на текст, який зазначено за умовчанням, щоб замінити його власним текстом).

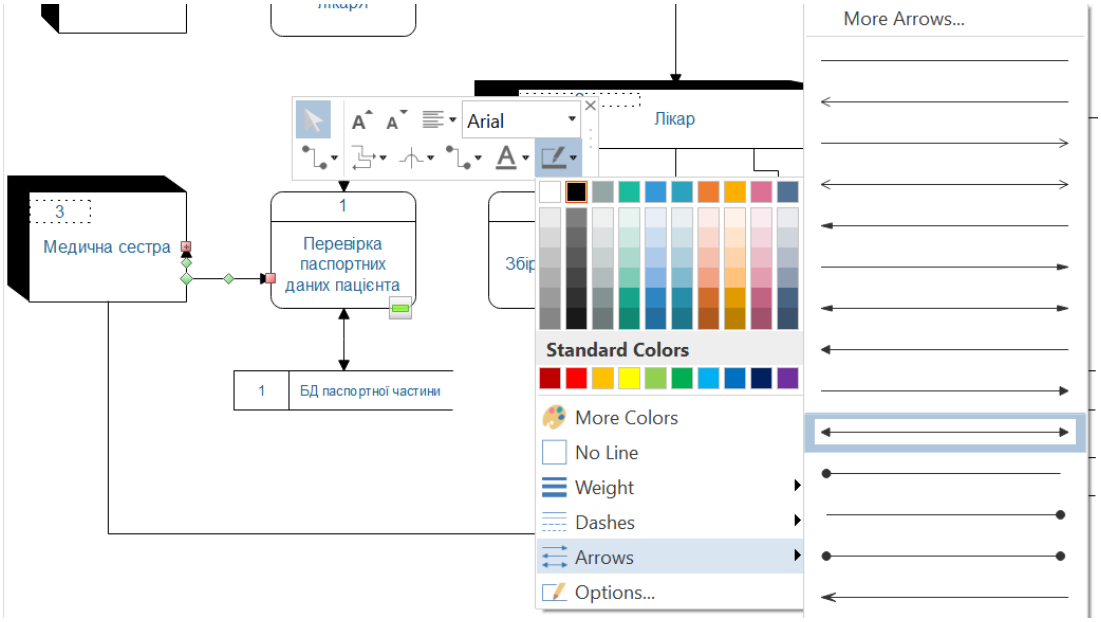


4. З’єднайте символи: використовуйте інтелектуальні з’єднувачі для з’єднання фігур. З’єднувачі будуть «приклеєні» до фігур.



5. При бажанні налаштуйте зовнішній вигляд: можна вибрати улюблений колір заливки для фігур та вибрати тему для схеми.

6. Для того, щоб поставити стрілку в обидва напрямки – виділяємо стрілку, натискаємо на вкладці «Home» розділ «Styles», далі Line, далі Arrows і вибираємо тип стрілки. Або за допомогою контекстного меню (див. рис. нижче)



Індивідуальне завдання

Засобами Edraw Max створити діаграму потоків даних при об’єктному (рис. 2.2) та функціональних (рис. 2.3) підходах для медичної документації згідно свого варіанту.

Приклад виконання побудови діаграми потоків даних для медичної документації форми 025/0 «Медична карта амбулаторного хворого»

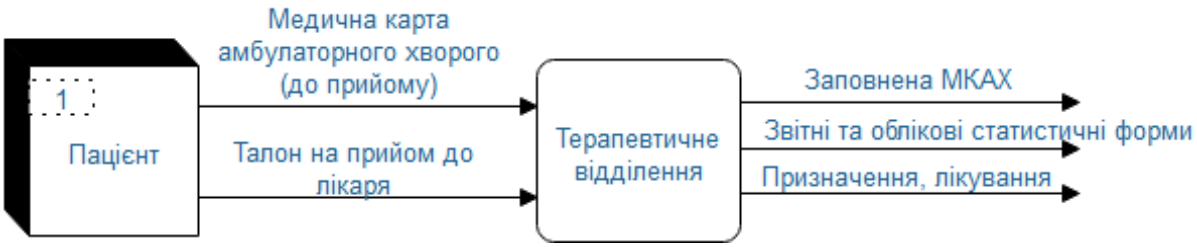


Рисунок 2.2 – Діаграма потоків даних при об’єктному підході (загальна схема)

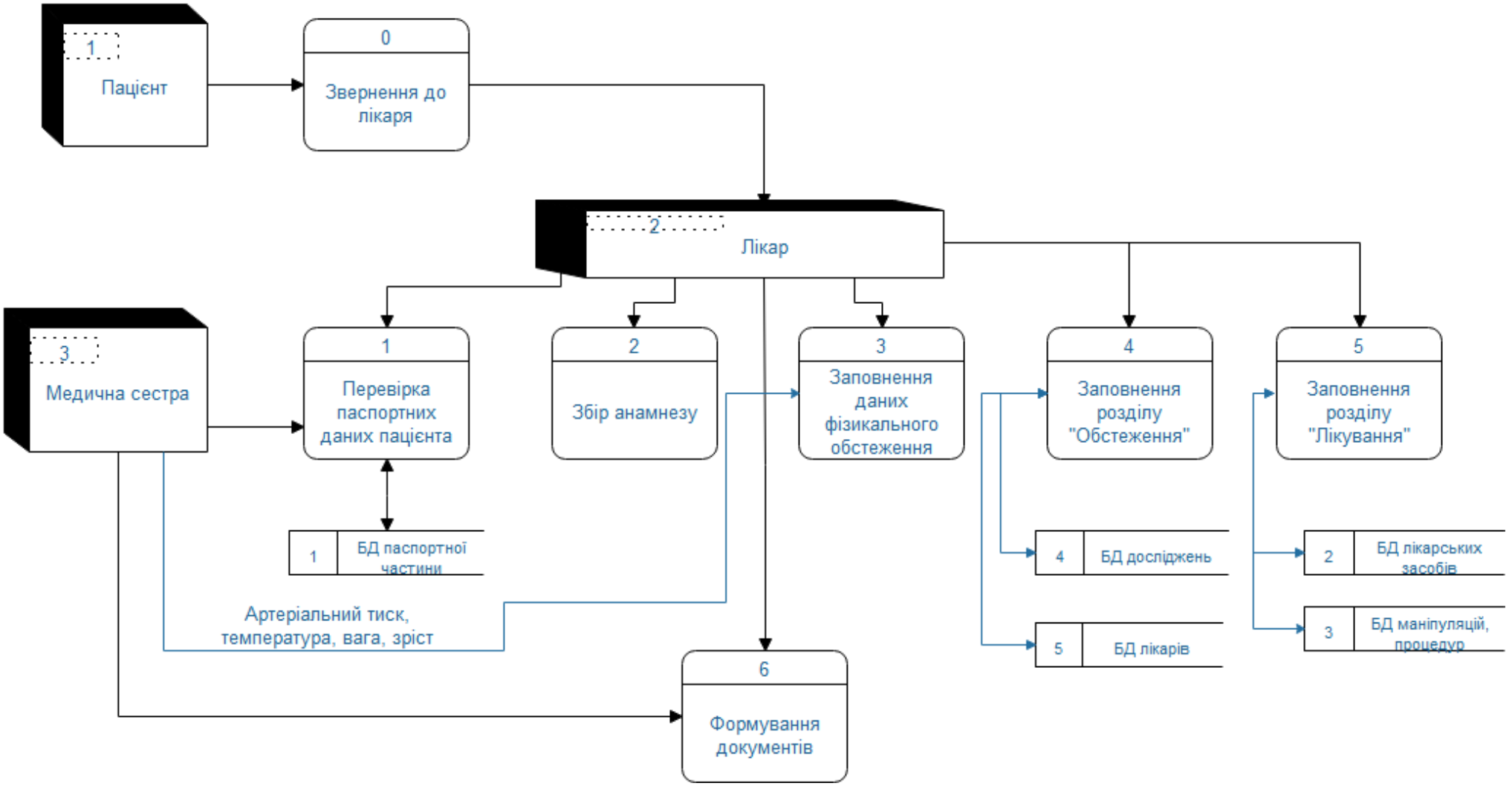


Рисунок 2.3 – Діаграма потоків даних при функціональному підході (деталізована схема)

Практична робота №3

Діаграма робочого процесу в нотації BPMN

Мета роботи – оволодіння прийомами моделювання робочого процесу в нотації BPMN для медичних облікових форм у програмі Edraw Max.

Теоретичні відомості

Для прозорості бізнес-процесів, а, отже, і спрощення розуміння та аналізу робочого процесу, використовується наочна візуалізація.

Робочий процес визначає спосіб виконання людьми поставлених задач, процесів та методів, які використовуються для того, щоб вся робота була завершена в уставлений час.

Діаграма робочого процесу – це візуальне представлення вказаного робочого процесу або бізнес-процесів та дій, таких як переміщення або передача даних, документів та задач протягом всієї кампанії, та проілюстрована за допомогою блок-схеми.

Для чого важливий робочий процес?

Робочий процес зручний для оптимізації, а також автоматизації різних повторюваних бізнес-операцій, що зводить до мінімуму помилки в процесі, тим самим підвищуючи загальну ефективність бізнес-моделей. Існують і інші переваги:

- Завдяки добре продуманому робочому процесі можна краще зрозуміти власні бізнес-процеси;
- Робочий процес допоможе виявити та усунути різні надмірності або непотрібні задачі, тим самим підвищуючи ефективність бізнесу;
- Добре спланований робочий процес полегшує співробітникам та персоналу розуміння того, що їм потрібно робити, тим самим знижуючи

рівень мікроменеджменту, який часто називають основною причиною звільнення;

- Наочність діяльності та підзвітність підвищуються за рахунок хорошого робочого процесу, що, в свою чергу, сприяє покращенню комунікації на робочому місці;

- І остання, але не менш важлива: добре спланований робочий процес може значно покращити якість продукту або послуги за рахунок усунення людських помилок, що, в свою чергу, допоможе бізнесу підвищити якість обслуговування клієнтів.

Стандарт BPMN (Business Process Model and Notation) розшифровується як «Нотація моделювання бізнес-процесів» та використовує блок-схему (рис. 3.1) дуже схожу на ту, яка використовується в UML-діях. Але BPMN виступає спільною мовою як для технічного, так і для ділового персоналу. BPMN фокусується не на результатах, а на різних внутрішніх процесах та інформації.

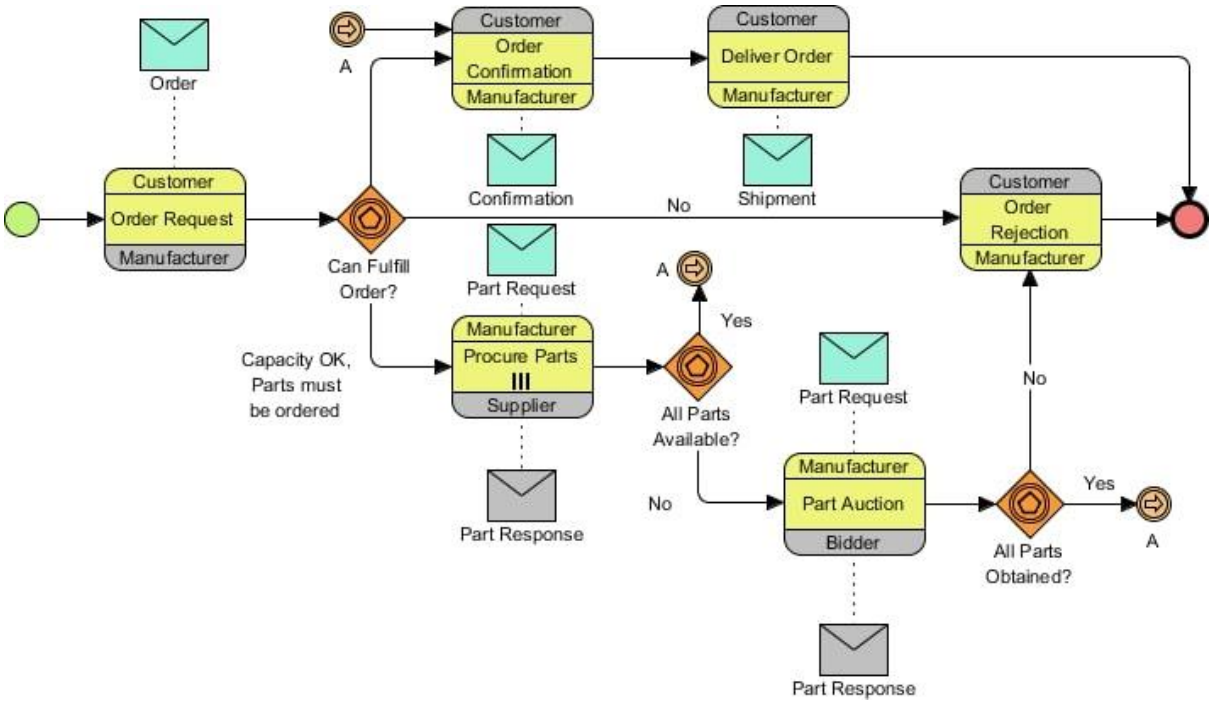


Рисунок 3.1 – Приклад бізнес-процесу у методології BPMN

Символ **Старту** сигналізує про перший етап процесу.

Символ **Кінця** позначає результат процесу.

Пул або **доріжка** можуть бути організацією, роллю або системою. Доріжки розділяють пули або інші доріжки ієрархічно.

Символ **задачі –** це одиниця роботи, яка представляє роботу, яку потрібно виконати.

Об'єкт даних представляє собою потік інформації в процесі, такої як ділові документи, електронні листи або паперові листи.

Введення даних – це зовнішнє введення для всього процесу.

Виведення даних – це результат всього процесу.

Підпроцес події поміщається в процес або підпроцес. Він активується, коли запускається його початкова подія, і може перервати контекст процесу більш високого рівня або працювати паралельно залежно від початкової події.

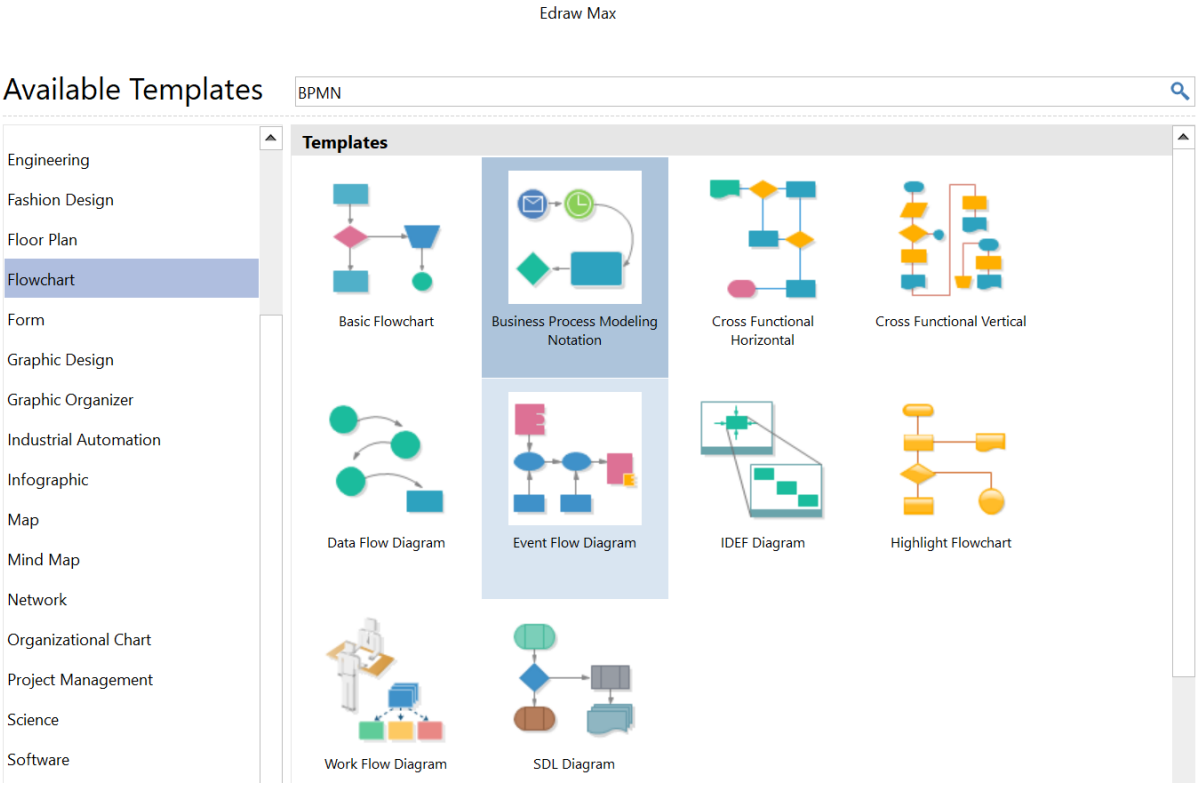
Посилання бесіди поєднує бесіди та учасників.

Асоціація даних використовується для зв'язкування елементів даних з діями, процесами і глобальними задачами.

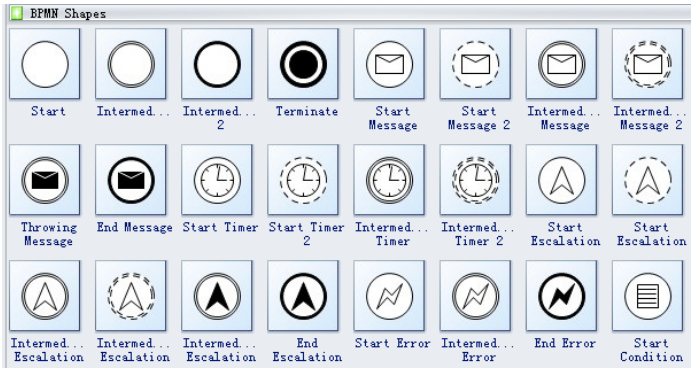
Сховище даних – це місце, де процес може читати або записувати дані, наприклад, база даних або шафа картотеки. Він зберігається поза часом існування екземпляру процесу.

Хід виконання роботи

1. Запустити Edraw Max. В розділі «Flowchart» вибрати «Business Process Modeling Notation»



2. Перенести необхідні елементи з лівою частини екрану на робочу область.



Індивідуальне завдання

Засобами Edraw Max створити діаграму діаграму робочого процесу у методології BPMN для медичної документації згідно свого варіанту.

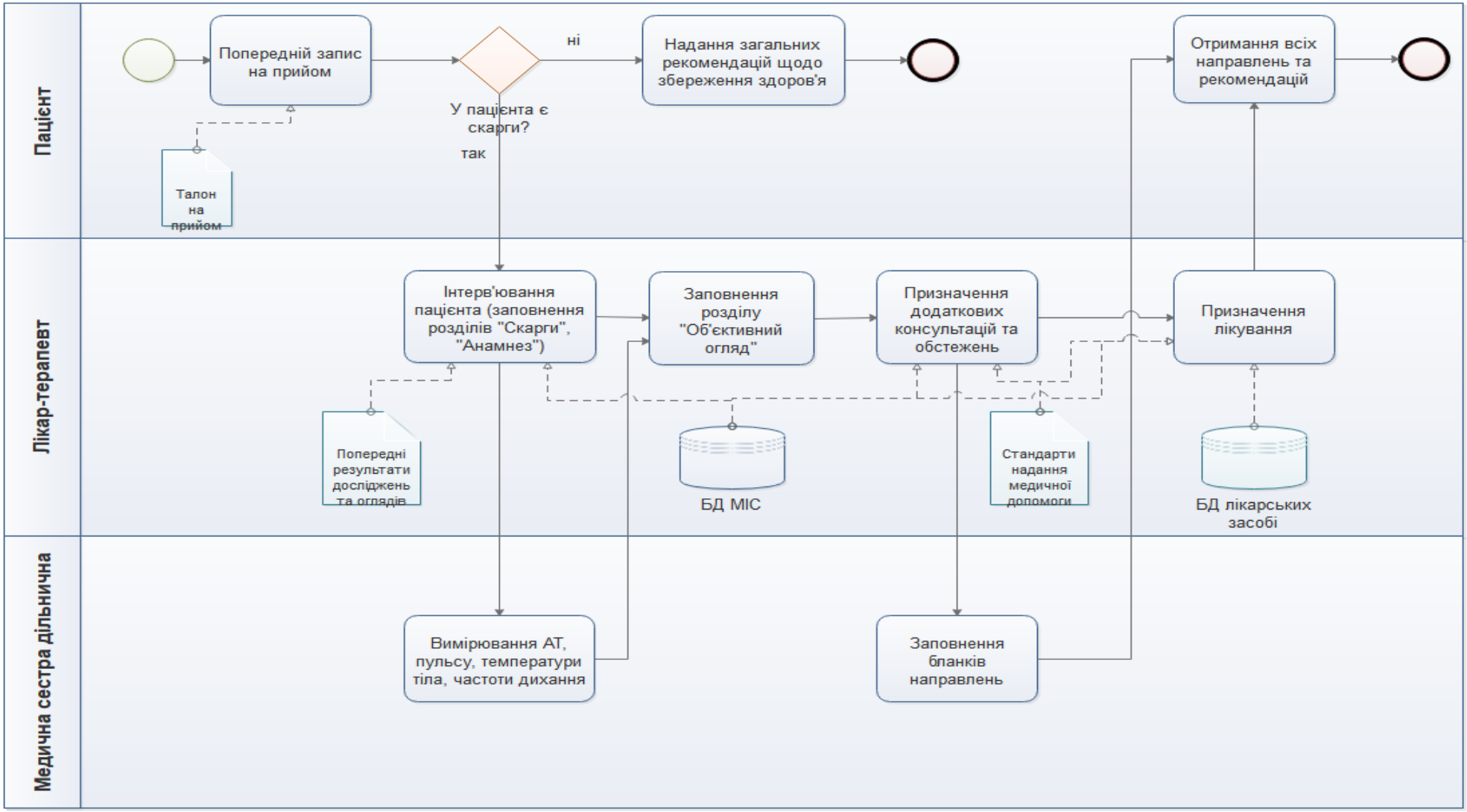


Рисунок 3.2 – Діаграма робочого процесу «Первинного відвідування лікаря пацієнтом з лікувально-діагностичною метою» (заповнення щоденника у формі 025/0 «Медична карта амбулаторного хворого»)

Практична робота №4

Діаграми UML

Мета роботи – оволодіння прийомами моделювання робочого процесу в нотації UML для медичних облікових форм у програмі Edraw Max.

Теоретичні відомості

UML (Unified Modeling Language) – це мова моделювання в області розробки програмного забезпечення.

Схема моделі UML допоможе описати та наглядно відобразити дизайн та структуру програмної системи.

Діаграма UML ідеально підходить для розробників програмного середовища та керівників проєктів, яким необхідно проілюструвати та інтерпретувати взаємозв'язки, дії та зв'язки додатків програмного забезпечення з використанням нотації Unified Modeling Language (UML).

Існуючі діаграми UML:

Діаграма варіантів використання UML (UML Use Case Diagrams). На ранніх етапах проєкта розробки для опису реальних дій та мотивацій. Схеми можна уточнити на більш пізніх етапах. На діаграмі відображається не окремий крок або транзакція, а відносно великий процес.

Діаграма статичної структури UML (UML Static Structure Diagrams). Діаграма, яка показує статичну структуру моделі, тобто існуючі елементи. Створені концептуальні діаграми представляють концепції реального часу та відношення між ними або діаграми класів, які розбивають програмне систему на частину.

Діаграма пакетів UML (UML Package Diagrams). Групування елементів моделі, яка представлена в UML символом, яка нагадує папку з

файлами. Кожний елемент в системі може відноситись лише до одного пакету, а один пакет може бути вкладений в інший. Один пакет може містити пакети, діаграми та інші елементи.

Діаграма діяльності UML (UML Activity Diagrams). Особливий вид діаграми станів, в якій всі стани є станами дій, а переходи закінчуються завершенням дій у вихідному стані. Діаграма діяльності використовується для опису внутрішньої поведінки методу та відобразити потік, який управляється внутрішніми діями.

Діаграма станів UML (UML Statechart Diagrams). Підкріплення кінцевого автомату прикріпленого до класу або методу, який описує реакцію класу на зовнішні стимули. Діаграма використовується, щоб показати послідовність станів, через які об’єкт проходить протягом свого існування.

Діаграма послідовності UML (UML Sequence Diagrams). На діаграмі зображуються актори та об’єкти, які взаємодіють, та події, які ними генеруються, впорядковані в часовій послідовності. Діаграма послідовності – це діаграма взаємодії, яка показує об’єкти, які беруть участь в конкретній взаємодії, та повідомлення, якими вони обмінюються, впорядковані в часовій послідовності. Актор на діаграмі представляє роль, яку грає зовнішній об’єкт. Таким чином, один фізичний об’єкт може бути представлений декількома діючими особами. Об’єктом є інформація, яка може бути створена в іншій програмі та імпортована, вбудована або зв’язана. Подією є будь-яка дія, наприклад, опитування пацієнта.

Діаграма співробітництва (кооперації) UML (UML Collaboration Diagram). Діаграма взаємодії, яка показує для однієї системної події, яка описується одним варіантом використання, як група об’єктів взаємодії між собою. Діаграма використовується для того, щоб показати взаємозв’язки між ролями об’єктів, наприклад, набір повідомлень, якими обмінюються учасники.

Діаграма компонентів UML (UML Component Diagrams). Діаграма реалізації, яка показує структуру коду. З діаграми компонентів можна дізнатись про залежність компілятора та часом виконання між програмними компонентами, такими як файл вихідного коду або бібліотеки DLL.

Діаграма розгортування UML (UML Deployment Diagrams). Діаграма реалізації, яка показує структуру системи з часом виконання.

Під час виконання комп'ютерного практикуму зупинимось на двох основних видах діаграм UML: діаграма варіантів використання та діаграма активності.

Діаграма варіантів використання UML використовується для представлення набору подій, які відбуваються, коли суб'єкт використовує систему для завершення процесу. Зазвичай зображується повний процес, а не окремий крок або транзакція. Можна робити уточнення на діаграмі на більш пізніх етапах, щоб відобразити інтерфейс користувача та деталі дизайну.

Діаграму варіантів використання UML використовують:

1. Розробники програмного забезпечення: представлення додатків з використанням нотації уніфікованої мови моделювання (UML).
2. Розробники програмного забезпечення: ілюструвати та інтерпретувати зв'язки, дії та зв'язки додатків програмного забезпечення.
3. Керівники програм: показувати статичні структури програмного забезпечення високого рівня в презентаціях та документації по специфікаціям.

Для діаграми Use Case:

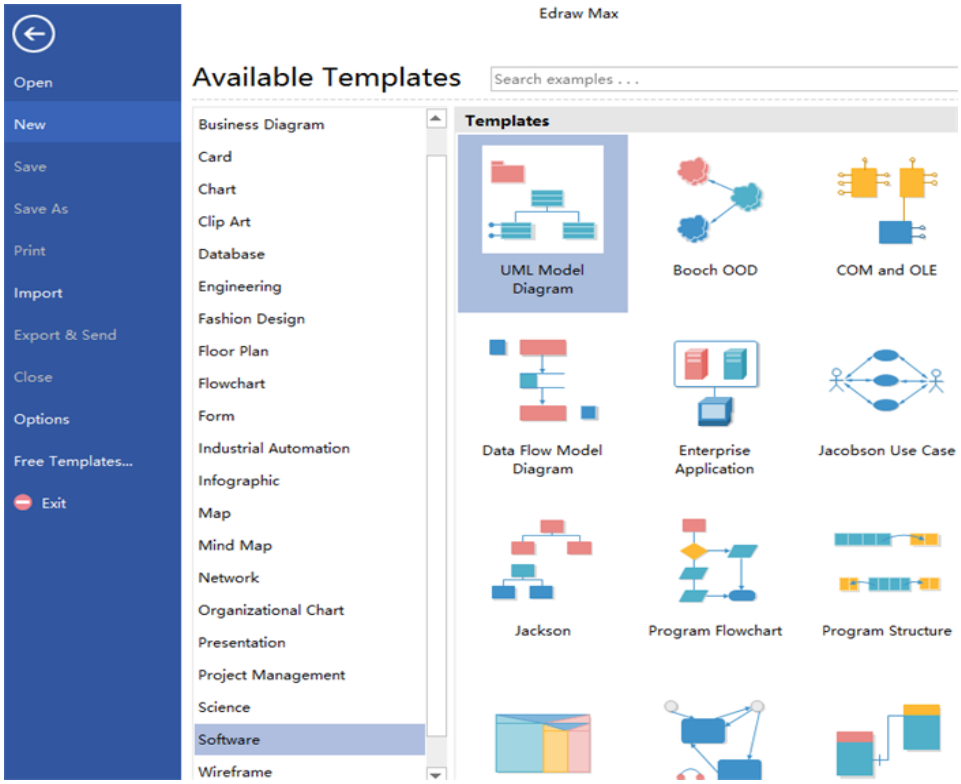
- ***Розширення (англ. Extend)*** - *різновид відносини залежності між базовим варіантом використання і його спеціальним випадком.*

- ***Включення (англ. Include)*** *- визначає взаємозв'язок базового варіанту використання з іншим варіантом використання, функціональне поведінка якого завжди* ***задіюється базовим варіантом*** *використання.*

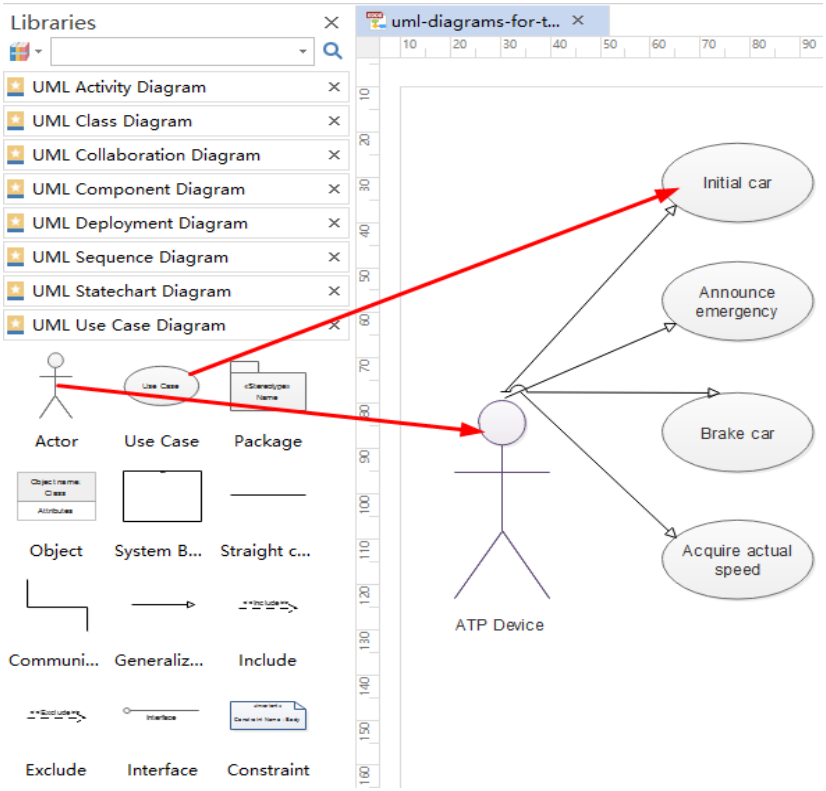
Діаграма активностей UML – це частковий випадок діаграми станів, в якій всі стани є станами дій, а переходи ініціюються завершенням дій в вихідному стані. Діаграма активностей використовується для опису внутрішньої поведінки методу та для представлення потоку, який управляється внутрішніми створеними діями.

Хід виконання роботи

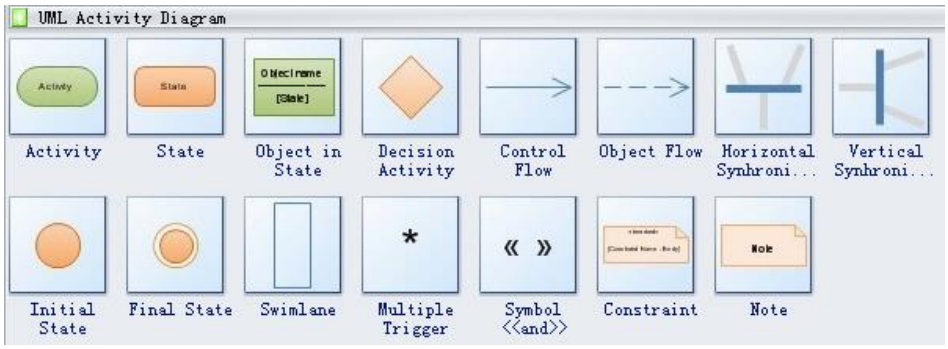
1. Запустити Edraw Max. В розділі «Software» вибрати «UML Model Diagram».



2. Перенести необхідні елементи з лівою частини екрану на робочу область (приклад для діаграми варіантів використання).



3. Перенести необхідні елементи з лівою частини екрану на робочу область (елементи діаграми активностей).



4. При підписанні стрілки користуйтеся звичайним текстовим блоком.

5. При побудові діаграми активностей змінити дизайн схеми, оскільки погано видно блоки початку та закінчення.

Індивідуальне завдання

Засобами Edraw Max створити діаграму робочого процесу у вигляді діаграми варіантів використання UML (рис. 4.1) та діаграми активностей UML (рис. 4.2) для медичної документації згідно свого варіанту.

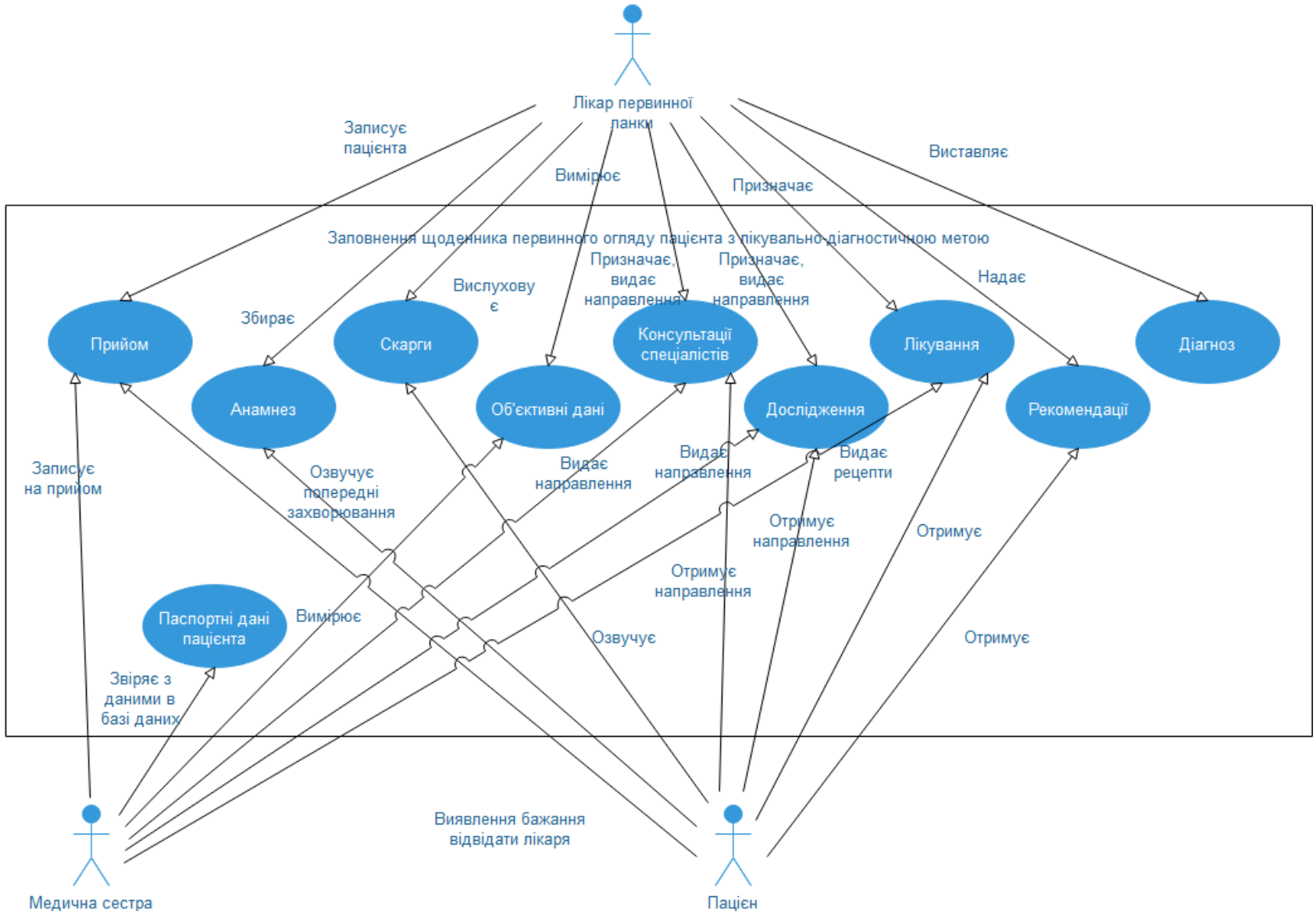


Рисунок 4.1 – Діаграма варіантів використання UML «Первинного відвідування лікаря пацієнтом з лікувально-діагностичною метою» (заповнення щоденника у формі 025/0 «Медична карта амбулаторного хворого»)

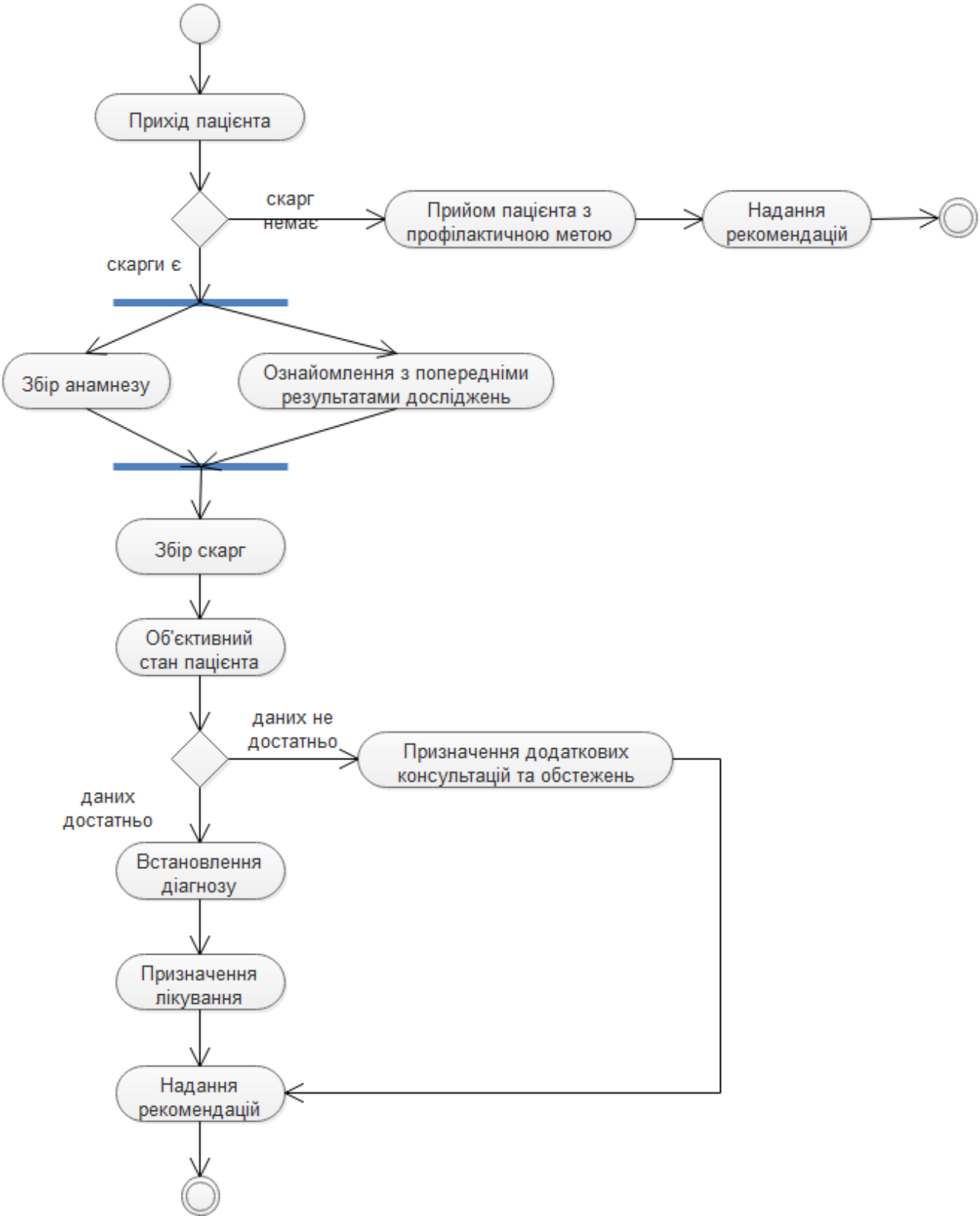


Рисунок 4.2 – Діаграма варіантів активностей UML «Первинного відвідування лікаря пацієнтом з лікувально-діагностичною метою» (заповнення щоденника у формі 025/0 «Медична карта амбулаторного хворого»)

Практична робота №5

Блок-схеми мовою ДРАКОН

Мета роботи – оволодіння прийомами моделювання робочого процесу мовою ДРАКОН для медичних облікових форм у програмі Fabula.

Теоретична частина

ДРАКОН – візуальна мова для зображення алгоритмів, процесів і процедур. Мета мови полягає в тому, щоб зробити процедури легкими для розуміння. Крім інформаційних технологій, ДРАКОН використовується в медичних алгоритмах. ДРАКОН знадобиться будь-де, де потрібно точно описати, як здійснити будь-яку процедуру. Також ДРАКОН допомагає керівникам організувати бізнес-процеси в своїх компаніях.

ДРАКОН можна визначити як загальнодоступну візуальну мову, яка призначена:

- Для систематизації, структуризації, наочного представлення та формалізації процедурних (імперативних) знань;
- Для опису структури людської діяльності;
- А також для проєктування, програмування, моделювання і навчання.

З ростом складності алгоритмів блок-схеми стрімко втрачають наочність, стають заплутаними і важкими для розуміння.

На відміну від них дракон-схеми відрізняються бездоганною *ясністю, прозорістю, чіткістю,* яка не залежить від складності.

Більш того, чим складніше алгоритми, тим більше виграш від мови ДРАКОН. Вона забезпечує швидкість і легкість розуміння за принципом: «Подивився - і відразу зрозумів!».

Завдяки використанню формальних і неформальних прийомів дракон-схеми дають можливість зобразити будь-який, як завгодно складний алгоритм (і будь-яку складну алгоритмічну систему, що складається з вкладених один в одного і паралельних алгоритмів) в наочній і зрозумілій формі (рис. 5.1).

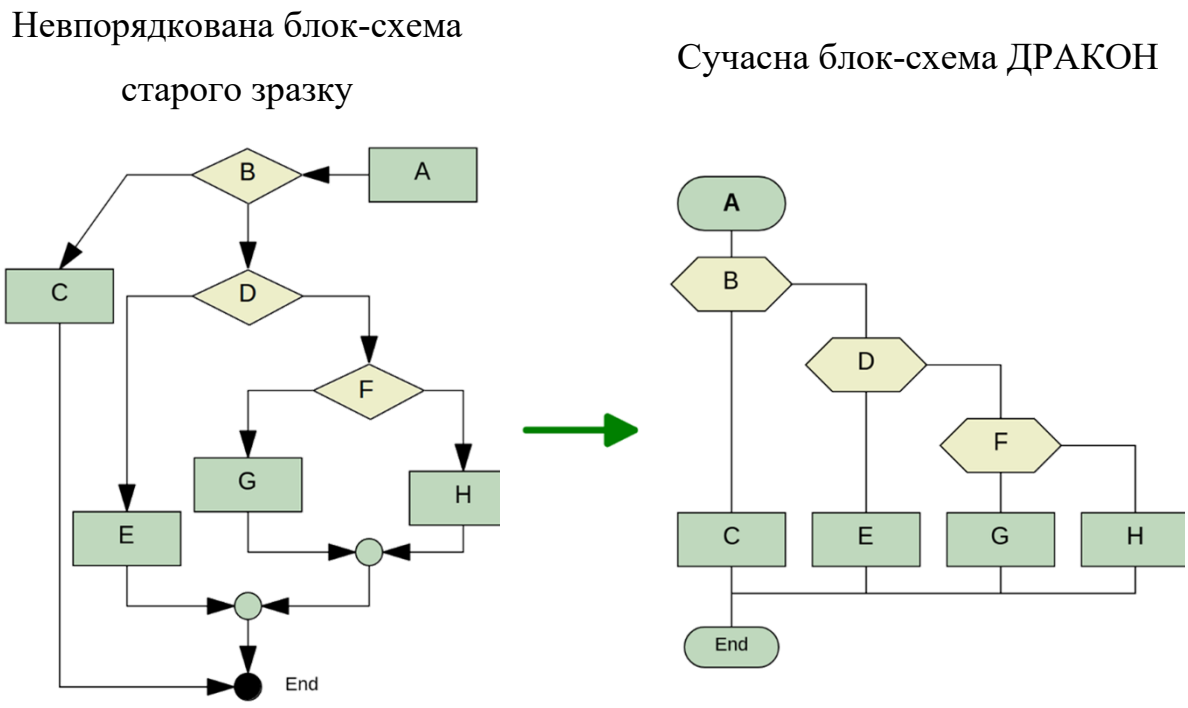


Рисунок 5.1 – Порівняння звичайних блок-схем та дракон-схем

Це дозволяє значно скоротити інтелектуальні зусилля персоналу, необхідні для розробки і перевірки алгоритмів і ієрархічних алгоритмічних систем, як формалізуються методом декомпозиції.

Блок-схеми не забезпечують автоматичне перетворення алгоритму в машинний код. Дракон-схеми, навпаки, придатні *для автоматичного отримання виконуваного коду*.

Головна перевага полягає в тому, що дракон-схеми (на відміну від блок-схем) *формуються методом логічного висновку за допомогою розробленого візуального логічного обчислення*, яке називається «обчисленням ікон»

Що робить ДРАКОН ефективним?

По-перше, ДРАКОН базується на *передових практиках* малювання блок-схем. Ось деякі з них:

- Перетин ліній заборонений;
- Дозволено лише прямі лінії та прямі кути;
- Замість стрілок використовуються прості лінії;
- Час на діаграмі «тече» зверху вниз, розгалуження іде вправо.

Ці та інші прийоми забезпечують одноманітність та зорову простоту діаграм.

По-друге, ДРАКОН має *унікальні властивості*, яких немає в інших візуальних мовах:

- *Шампур* підсвічує на діаграмі «царську дорогу» (happy path);
- *Силует* розбиває діаграму на логічні частини та допомагає долати складнощі;
- *Загальна доля* показує неявні зв'язки між діями, які знаходяться на різних шляхах.

Все це разом дає ДРАКОНу якісну перевагу над іншими графічними нотаціями.

Розглянемо, з чого ж складається дракон-схема.

Головними ознаками дракон-схем є те, що у верхній частині схеми міститься так звана «шапка» схеми, а знизу – адреси та закінчення схеми (рис. 5.2).

Шапка дракон-схеми – надзвичайно ефективний інструмент, який забезпечує структуризацію дракон-схеми та розбиття алгоритму на смислові частини – гілки.

Ергономічна хитрість полягає в тому, що шапка, вгадуючи потаємне бажання читача, дає йому сильну підказку – відповідь на всі «царські» питання

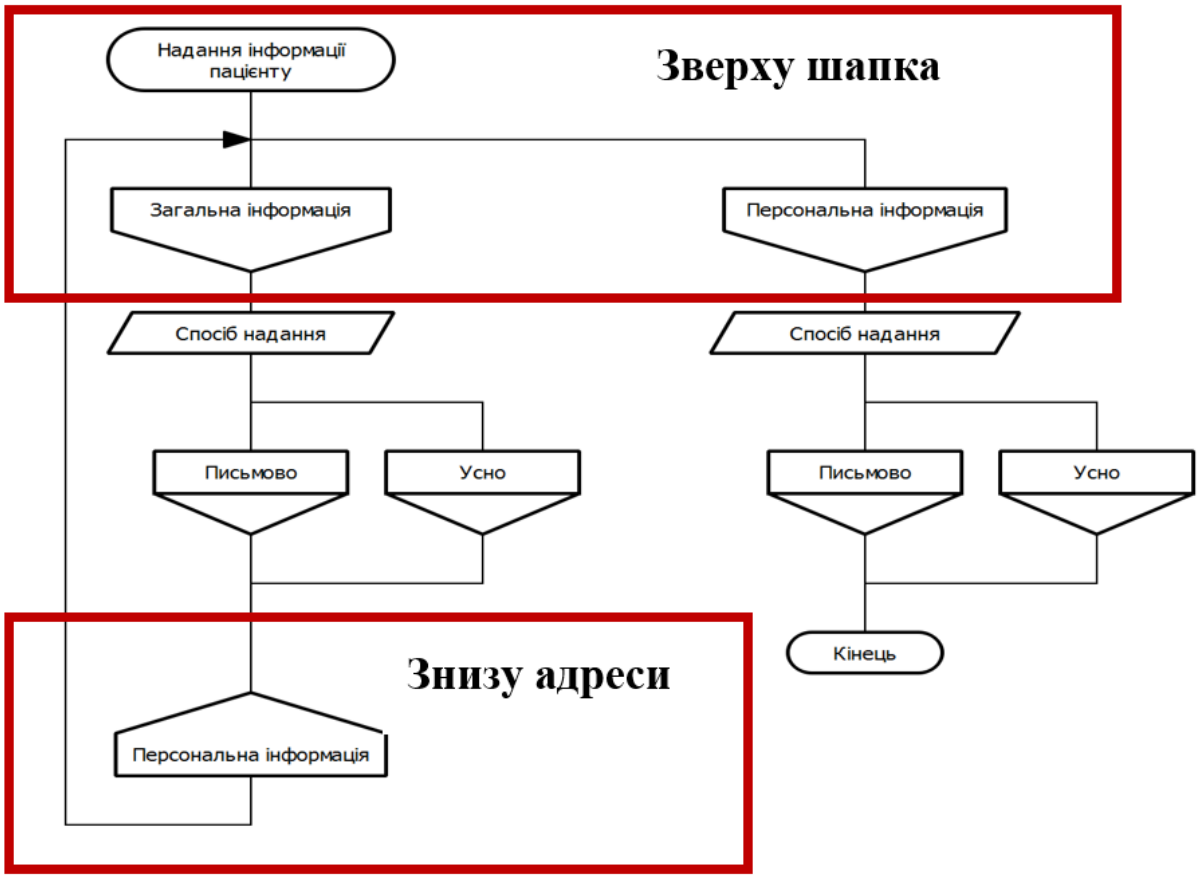


Рисунок 5.2 – Дракон-схема

Шапка дракон-схеми відповідає на три «царських» питання:

- Як називається проблема? *(Читаємо заголовок алгоритму)*
- З скількох частин вона складається? *(Рахуємо ікони «Ім'я гілки»)*
- Як називається кожна частина? *(Читаємо текст в іконах «Ім'я гілки»)*.

В мові ДРАКОН використовується два дуже зручних, але незвичних поняття.

Шампур – вертикальна лінія, яка з'єднує початок та кінець алгоритму (або гілки).

Головний маршрут – шлях від початку до кінця алгоритму, який веде до найбільшого успіху.

Правило говорить: *головний маршрут алгоритму повинен іти по шампуру* (рис. 5.3). Це означає, що «царський» маршрут не може виявитись десь на задвірках дракон-схеми, де його неможливо знайти. Ні, він повинен завжди знаходитися на «найпочеснішому» місці – крайній лівій вертикалі. Дотримання цього правила робить схему візуально впорядкованою, передбачуваною та інтуїтивно ясною.

Рисунок 5.3 – Помилки при побудові головного маршруту

Як же правильно має виглядати схема?

Є два варіанти побудови (рис. 5.4). З логічної точки зору обидва алгоритми еквівалентні. Але з ергономічної точки зору ліва схема краща, оскільки головний маршрут не має зломів.

Також згідно правил побудови дракон-схем є важливий нюанс: найкращий варіант розвитку подій зображується зліва першим, найгірший – останнім справа (рис. 5.5).

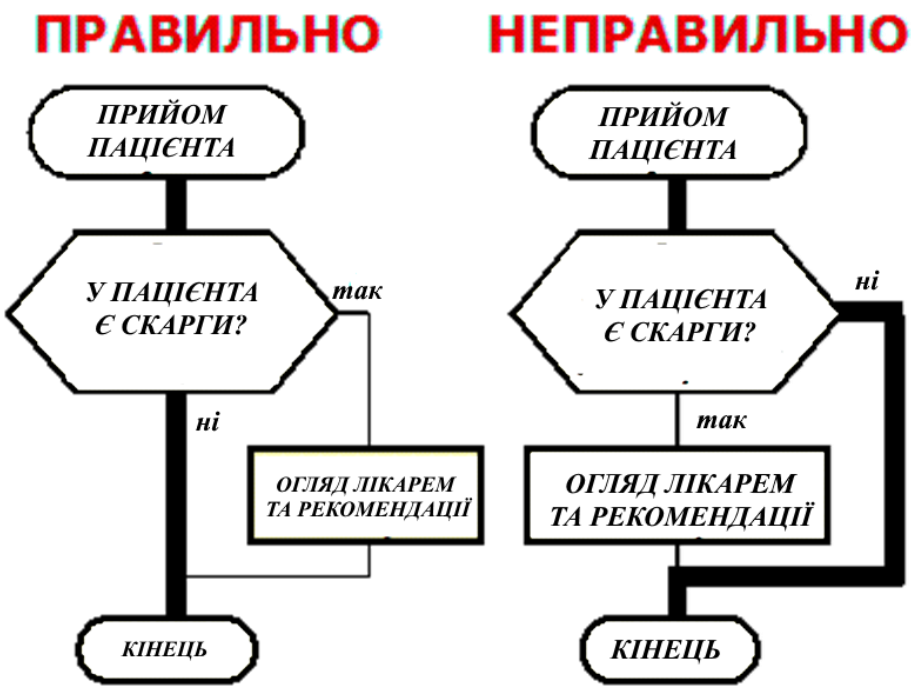


Рисунок 5.4 – Правильність побудови схеми

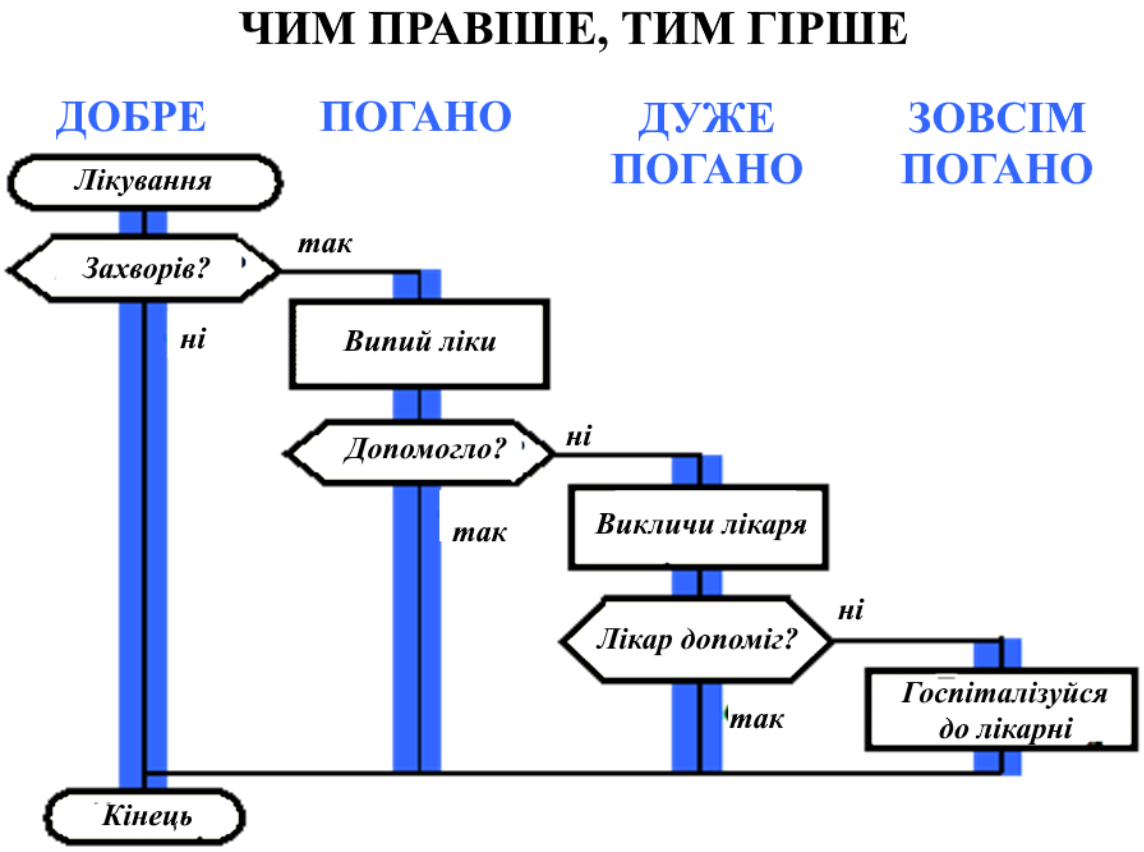


Рисунок 5.5 – Дракон-схема впорядкована зліва направо згідно принципу «Чим правіше, тим гірше»

Ікони мови ДРАКОН

Щоб графічний мова стала гнучкою і виразною, потрібно ретельно спроектувати графічний алфавіт мови, зробити його ергономічним.

Графоелементи (графічні літери) мови ДРАКОН називаються іконами (рис. 5.6). У мові ДРАКОН 26 ікон.

Ікона	Назва ікони
	Заголовок
	Кінець
	Дія
	Питання
	Вибір
	Варіант
	Ім’я гілки
	Адреса
	Вставка
	Полиця
	Формальні параметри
	Початок циклу ДЛЯ
	Кінець циклу ДЛЯ

Ікона	Назва ікони
	Виведення
	Введення
	Пауза
	Період
	Пуск таймеру
	Синхронізатор (за таймером)
	Паралельний процес
	Коментар
	Правий коментар
	Лівий коментар
	Петля циклу
	Петля силуету
	З’єднувач

Рисунок 5.6 – Ікони мови ДРАКОН

Як намалювати блок-схему?

Почни з відповідної назви. Перед тим, як почати малювати, придумайте відповідну назву для діаграми. Назва повинна точно відображати призначення алгоритму або процедури. Вона повинно бути якомога коротшою, але не занадто короткою. Назва повинна нести сенс.

Помісти початок вгорі. Початкову ікону слід помістити нагорі діаграми. Саме там читач і буде її шукати. Давайте не будемо витрачати час читача даремно і дамо йому те, що він очікує.

Початкова ікона повинна містити назву діаграми. Це не повинно бути слово "початок" або "почни тут".

Кінець тільки один. На діаграмі повинен бути тільки один кінець. Ікону Кінець слід помістити внизу діаграми. Дайте читачеві почуття безпеки: щоб не сталося на діаграмі, все закінчиться там, де має.

В іконі Кінець повинен бути напис "Кінець". Це сигнал для читача: тепер все скінчено. Не розміщуйте у ікону Кінець останній крок алгоритму або назву наступної процедури.

Йди вниз. Потік виконання на діаграмі повинен йти зверху вниз. Цей напрямок - найзручніший, так як люди звикли читати тексти таким чином. А крім того, рух вниз природно на планетах з силою тяжіння.

Уникай поворотів. Єдиний випадок, коли лінії повинні змінювати напрямок, - це прийняття рішень. Коли є вибір між декількома шляхами, ці шляхи повинні спочатку розійтися, а потім знову зійтися. Для цього, звичайно, потрібні повороти.

Якщо рішень немає, не повертайте. Ідіть вниз.

Якщо рішення є, мінімізуйте число поворотів.

Не допускай перетину ліній. Всякий раз, коли око натикається на перетин ліній, наш мозок намагається з'ясувати, а чи не пов'язані ці лінії? Це створює додаткове навантаження на мозок.

Всі спроби зображувати перетин так, щоб це не виглядало жахливо, провалилися. Єдиний спосіб уникнути цього додаткового навантаження - не допускати перетинів.

Заміни стрілки простими лініями. За старих часів блок-схеми складалися з квадратиків і стрілок. Зараз це вже не так.

В сучасних ДРАКОН-схемах замість стрілок є прості лінії. Чому? Справа в тому, що з квадратиками проблем немає, а ось зі стрілками - є. Стрілки являють собою додаткові графічні об'єкти, і вони збільшують складність візуальної сцени. Призначення стрілки - показати наступну ікону. А якщо ми приймаємо умову про те, що ікони йдуть зверху вниз, то стрілки стають взагалі не потрібні! Наступна ікона завжди під поточною.

Є тільки одна ситуація, коли наступна ікона розташована вище поточної. Це можливо, коли є цикл. Тільки в тому випадку має сенс використовувати стрілку.

Таким чином, цикли стають добре видно на блок-схемі: шукайте стрілки!

Тільки прямі вертикальні і горизонтальні лінії Вигнуті лінії можуть бути доречні, коли мова йде про дизайн. Але в діловій графіці криві - це отрута. Ось чому. Наш мозок розглядає відрізки прямих як прості примітиви. Ми легко бачимо, які два об'єкти з'єднані прямим вертикальним або горизонтальним відрізком.

З кривими і похилими лініями все по-іншому. Щоб зрозуміти, які саме об'єкти з'єднує крива лінія, око змушене ретельно відстежити цю лінію від початку до кінця. Це створює непотрібну напругу всередині нас і займає час.

Вирівняй ширину ікон на вертикалі. Коли кілька ікон на одній вертикалі мають одну і ту ж ширину, ми сприймаємо їх як групу. Наші очі сканують їх швидше. Але якщо у ікон різна ширина, відчуття м'якого ковзання від однієї ікони до іншої втрачається.

Коли ширина ікони відрізняється від ширини сусідів, це сигнал. Не давайте читачеві хибних сигналів.

Дотримуйся однакової відстані між сусідніми елементами. Що таке "метр"? У поезії метр - це базова ритмічна структура строфи.

У графіку метром називають вимогу дотримуватися однакової відстані між сусідніми елементами. Метр - це простий трюк, але його позитивний ефект величезний.

Розгалуження: тільки вправо! Ми домовилися про те, що наші блок-схеми будуть виконуватися зверху вниз. Нам залишається тільки два напрямки для додаткових шляхів, які починаються в точках прийняття рішень: ліво і право. На практиці виявляється, що корисно вибрати тільки один напрямок і дотримуватися його.

Завжди направляйте додаткові шляхи тільки вправо. Виконання цього правила істотно підвищує передбачуваність діаграм і їх однаковість. Але ж передбачуваність - необхідна умова ясності.

Читач не повинен сканувати діаграму в пошуках додаткового шляху. І так відомо, що він праворуч. Читач очікує розгалуження справа і знаходить його там.

Цей нехитрий прийом економить чимало сил. Замість того, щоб спочатку аналізувати форму діаграми, можна відразу перейти до її змісту.

Чим правіше, тим гірше. Не всі шляхи через діаграму однакові. Зазвичай один з них найбільш успішний. Такий шлях називають "царська дорога" (або happy path). Решта маршрутів в залежності від контексту якимось чином гірші. У програмуванні деякі умови призводять до помилок і збоїв. У медичних процедурах існує можливість того, що пацієнт помре.

У деяких алгоритмах важко сказати, чи є який-небудь результат хорошим або поганим. У таких випадках царська дорога - це найбільш вірогідний сценарій.

Існує простий спосіб чітко позначити царську дорогу на блок-схемі. Блок-схеми слід малювати так, щоб **царська дорога проходила по головній вертикалі**. Вертикаль з царської дорогою називається "шампур".

Коли через діаграму проходить більше одного маршруту, відсортуйте їх зліва-направо відповідно до принципу: чим правіше, тим гірше. Царська дорога буде на самій лівій вертикалі, найгірший сценарій - на самій правій. Всі інші шляхи пройдуть десь в середині.

Ті читачі, які хочуть знати тільки царську дорогу, не зобов'язані вивчати всю діаграму. Їм досить кинути погляд тільки на ліву її частину.

Фактичне використання ДРАКОН в медицині

Настільки багатий алфавіт має велику виразною силою. Він дозволяє зобразити алгоритми і ієрархічні алгоритмічні системи будь-якої складності, включаючи паралельні процеси і процеси реального часу. І забезпечити максимальну наочність і зрозумілість отриманої «картинки», тобто математично строгого комплекту ергономічних алгоритмічних креслень.

Альгірдас Каралюс (Литва) був першим, хто звернув увагу на можливість великомасштабного використання мови ДРАКОН в медицині.

Ініціативу Каралюса активно підтримали багато литовські лікарі. За останні рік в Литві видані три медичних підручника, в яких використовується ДРАКОН:

1. Початкова невідкладна акушерська допомога.
2. Невідкладна медична допомога.
3. Спеціалізована реанімація новонародженого.

Підручники призначені для медичних працівників швидкої допомоги, фахівців, що працюють в приймальних відділеннях і учнів - майбутніх акушерів, фельдшерів і студентів медичних вузів.

Індивідуальне завдання

Засобами FABULA розробити блок-схему мовою ДРАКОН для свого варіанту (відповідної медичної форми облікової документації).

Приклад реалізації наведено на рисунку 5.7.

Хід виконання роботи

1. Розпакувати архів FABULA в будь-яке зручне місце на жорсткому диску.
2. Запустити на виконання файл fabula_0.1b_r001.exe.
3. Перенести необхідні елементи з лівою частини екрану на робочу область.

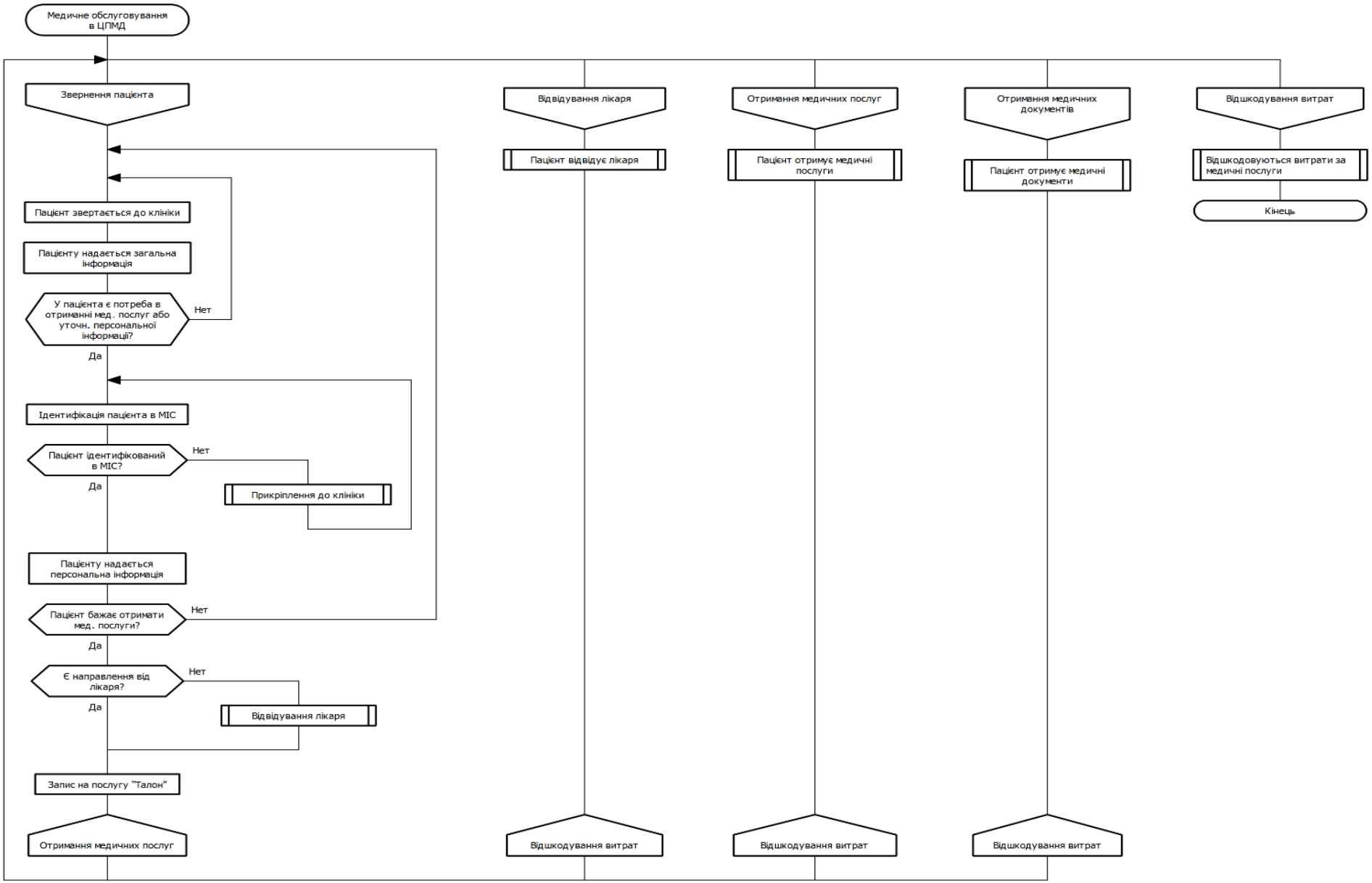


Рисунок 5.7 - Блок-схема мовою ДРАКОН процесу «Медичне обслуговування в Центрі первинної медичної допомоги»

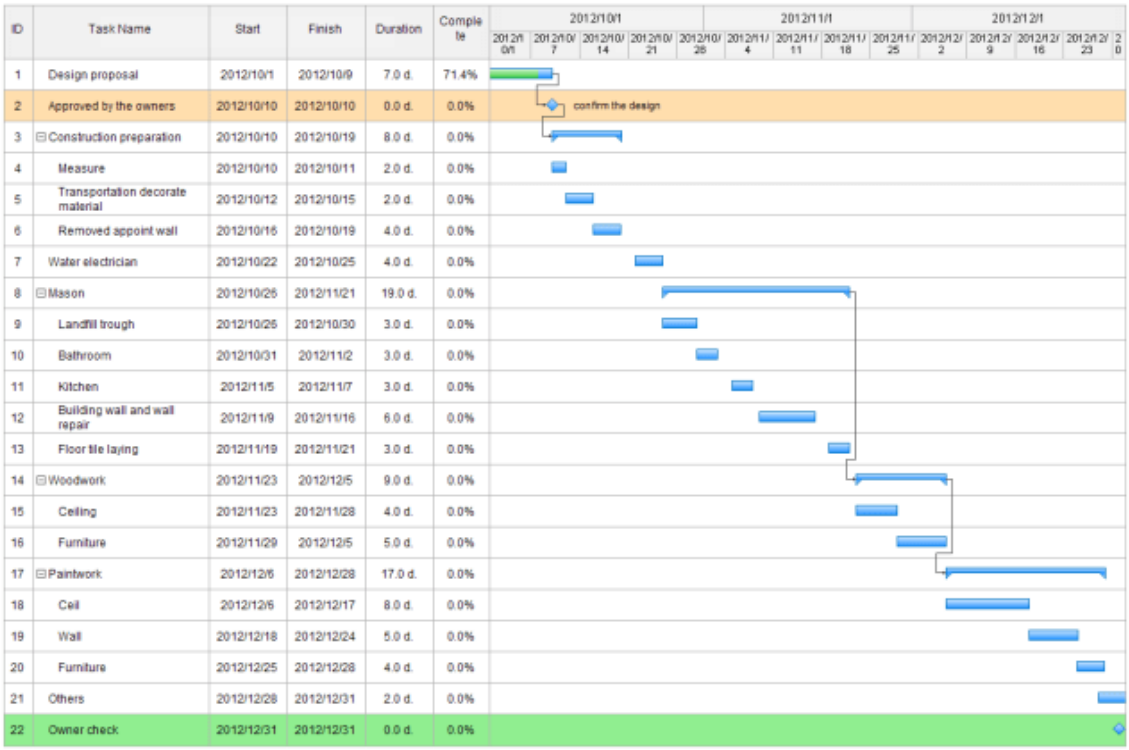
Практична робота №6

ДІАГРАМА ГАНТА ПРИ МОДЕЛЮВАННІ БІЗНЕС-ПРОЦЕСІВ

Мета роботи – оволодіння прийомами створення інструменту управління проєктом у вигляді діаграми Ганта у програмі Edraw Max.

Теоретична частина

Діаграма Ганта – це різновид кольорової смуги візуальної презентації серії завдань із розбиттям за певний період (рис. 6.1). Горизонтальна вісь представляє загальний проміжок часу та послідовність завдань на основі щоденної, тижневої чи місячної частоти, тоді як вертикальна вісь показує конкретні завдання, які необхідно виконати. Керівники проектів використовують такі діаграми для оновлення та моніторингу ресурсів та членів команди на шляху до досягнення кінцевої мети.



ID	Task Name	Start	Finish	Duration	Complete
1	Design proposal	2012/10/1	2012/10/9	7.0 d.	71.4%
2	Approved by the owners	2012/10/10	2012/10/10	0.0 d.	0.0%
3	Construction preparation	2012/10/10	2012/10/19	8.0 d.	0.0%
4	Measure	2012/10/10	2012/10/11	2.0 d.	0.0%
5	Transportation decorate material	2012/10/12	2012/10/15	2.0 d.	0.0%
6	Removed appoint wall	2012/10/16	2012/10/19	4.0 d.	0.0%
7	Water electrician	2012/10/22	2012/10/25	4.0 d.	0.0%
8	Mason	2012/10/26	2012/11/21	19.0 d.	0.0%
9	Landfill trough	2012/10/26	2012/10/30	3.0 d.	0.0%
10	Bathroom	2012/10/31	2012/11/2	3.0 d.	0.0%
11	Kitchen	2012/11/5	2012/11/7	3.0 d.	0.0%
12	Building wall and wall repair	2012/11/9	2012/11/16	6.0 d.	0.0%
13	Floor tile laying	2012/11/19	2012/11/21	3.0 d.	0.0%
14	Woodwork	2012/11/23	2012/12/5	9.0 d.	0.0%
15	Ceiling	2012/11/23	2012/11/28	4.0 d.	0.0%
16	Furniture	2012/11/29	2012/12/5	5.0 d.	0.0%
17	Paintwork	2012/12/6	2012/12/28	17.0 d.	0.0%
18	Ceil	2012/12/6	2012/12/17	8.0 d.	0.0%
19	Wall	2012/12/18	2012/12/24	5.0 d.	0.0%
20	Furniture	2012/12/25	2012/12/28	4.0 d.	0.0%
21	Others	2012/12/28	2012/12/31	2.0 d.	0.0%
22	Owner check	2012/12/31	2012/12/31	0.0 d.	0.0%

Рисунок 6.1 – Приклад діаграми Ганта

Діаграма Ганта, яка зображена горизонтально та візуально, пояснює загальний проект, його поточний стан, його прогрес та людей, які беруть участь у ньому. Типова діаграма Ганта детально описує всі етапи процесу разом із датами їх початку та завершення. Горизонтальні смуги на кресленні представляють прогрес, а сполучні лінії між двома етапами показують, коли завдання було перенесено з фази на наступну.

Чому використовуються діаграми Ганта - переваги та недоліки

Діаграма Ганта показує всі завдання, які необхідно вирішити для завершення проекту разом із очікуваним часом виконання. Деякі з основних переваг, які дає діаграма Ганта, включають:

- **Планування.** Діаграма Ганта дозволяє менеджерам мати приблизні орієнтири на початковій стадії проекту. Діаграма показує, скільки фаз буде в процесі і хто буде відповідати за якесь завдання.
- **Розподілення ресурсів.** Діаграма Ганта показує, які ресурси будуть потрібні в проекті та як і в якій мірі вони будуть задіяні, щоб максимальний видобуток можна було отримати за мінімальних інвестицій.
- **Моніторинг та оновлення.** Як менеджер, ви навіть можете час від часу відстежувати хід виконання проектів і за допомогою діаграми Ганта отримати чітке уявлення про поточний стан доступності ресурсів.
- **Поправки.** Оскільки проекти можуть йти не так, як вони були заплановані спочатку, і завжди потрібно вносити деякі зміни, коли справа йде на практичних підставах, діаграма Ганта чітко уявляє невідповідності, через які може проходити процес. З таким розумінням керівникам стає легше вирішувати проблеми, вносячи основні зміни в робочий процес.
- **Швидко розмістити «Великі та маленькі фотографії».** Різні ролі можуть мати різну потребу з точки зору знання того, що відбувається з конкретним проектом. "Велика картина" допоможе людям, які не мали досвіду учасника проекту, наприклад зацікавлених сторін чи інших зовнішніх спонсорів, швидко пройти загальний відсоток виконання або

процес конкретного завдання. Навпаки, подання "невелика картинка" призначене для членів команди проекту, які потребують більш детальної перевірки свого проекту.

- **Більше відповідального спілкування з командою.** Призначення завдань, взаємозв’язки завдань та прогрес проекту є достатньо чіткими, щоб керівники команд могли контролювати роботу та змушувати розслаблених учасників прискорюватися. Більш того, віхи в діаграмах Ганта можуть служити мотиваційним інструментом.
- **Краще використання ресурсів.** Діаграми Ганта чудово підходять для вирішення проблем жонглювання кількома різними проектами. Крім того, для завдань, які мають одночасну тривалість, ви як керівник проекту можете легко відстежувати та виявляти перевантажені або недостатньо використані завдання за допомогою діаграм Ганта.
- **Оцінюйте завдання більш відповідним чином.** Оцінка проекту здається великим викликом для керівників проектів. Однак у діаграмах Ганта керівники команд можуть оцінювати на основі багатих наявних даних та ресурсів.

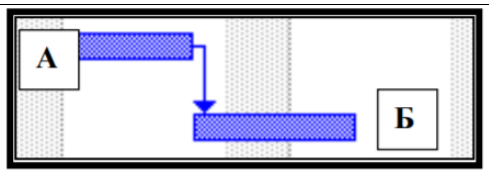
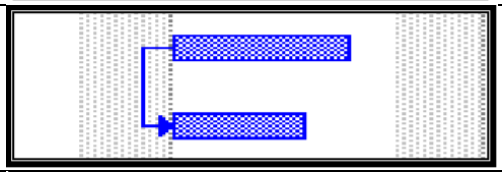
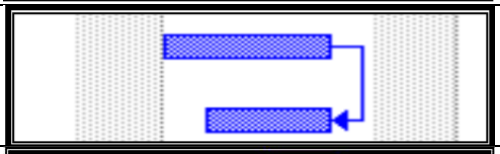
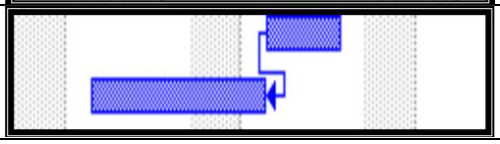
Як і у всьому іншому, використання діаграми Ганта також має деякі мінуси. Нижче перераховані ці недоліки:

- **Недосконалість.** Оскільки діаграми Ганта містять майже всі, якщо не всі, деталі проекту, вони з часом можуть стати занадто складними і в кінцевому підсумку втратити досконалість. Простими словами, коли йдеться про точність, на діаграми Ганта можна розраховувати, але не можна вважати їх повністю надійними.
- **Відносність.** Діаграми Ганта не завжди чітко показують відповідні завдання, наприклад, ті, які залежать від ілюстрованих. Простіше кажучи, діаграма Ганта хороша лише для демонстрації основних завдань, які необхідно виконати для завершення проектів, і не містить усіх деталей процесу.

Типи зв’язків між задачами

У процесі складання плану роботи дуже важливо визначити залежності (зв'язки) між завданнями. Як правило, залежність встановлюється між датою закінчення однієї задачі та датою початку другої. Зв’язки між завданнями проекту можуть бути більш складними. У таблиці 6.1 наведено типи зв’язків між задачами, які можуть використовуватись при створенні діаграми Ганта

Таблиця 6.1 – Типи зв’язків

Тип залежності	Представлення на діаграмі Ганта	Опис
Закінчення-початок (ЗП)		задача Б не може розпочатись, поки не закінчиться задача А
Початок-початок (ПП)		задача Б не може розпочатись, поки не розпочнеться задача А
Закінчення-закінчення (ЗЗ)		задача Б не може закінчитись, поки не закінчиться задача А
Початок-закінчення (ПЗ)		задача Б не може закінчитись, поки не розпочнеться задача А

Що є складовими діаграми Ганта?

Яким би складним не був проект, діаграму Ганта можна розбити на п'ять основних частин:

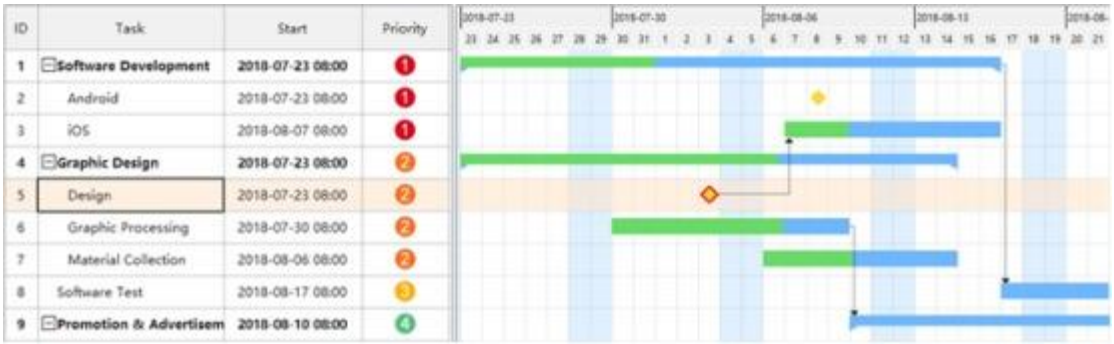
1. Назви завдань - це безпосередньо показує, про що йде мова, наприклад, цифровий дизайн або розробка програмного забезпечення тощо.

2. Панелі завдань - цей елемент вказує таку інформацію, як дата початку та завершення завдання, тривалість завдання та хід виконання у

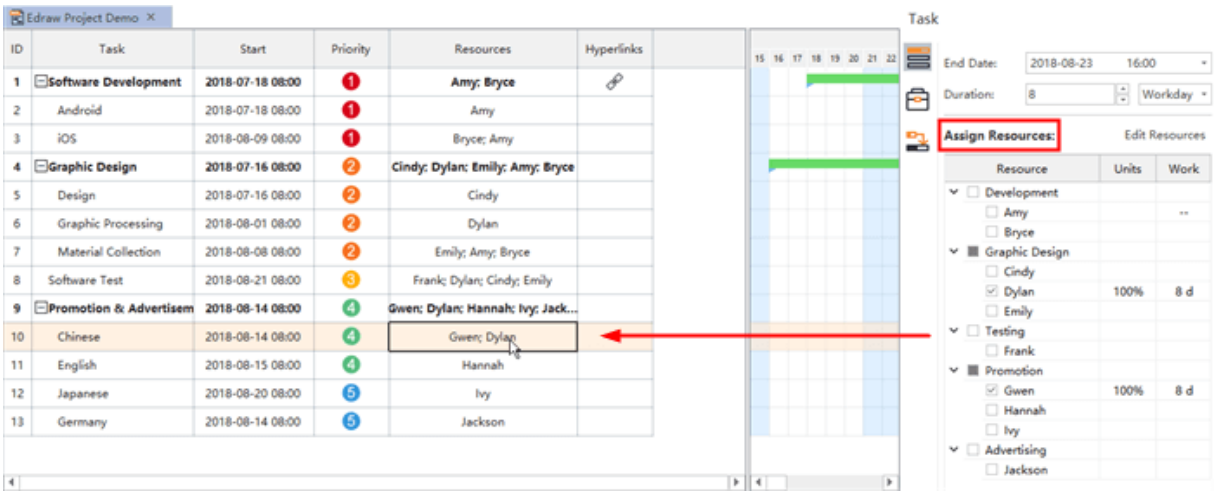
відсотках. Для виконання завдання використовуються два різні кольори, щоб відрізняти завершену частину та незавершену частину.

3. Взаємовідносини завдань - зазвичай вони охоплюють пріоритет та залежності завдань, наприклад, наскільки одна дія перекриває іншу, або послідовність виконання завдань; Як правило, залежності завдань можна встановлювати на основі початку до кінця, від початку до початку, від кінця до кінця та від початку до початку. Залежність від початку до початку означає, що завдання B не може розпочатися, доки завдання A не буде завершено.

4. <u>Основні етапи</u> , які часто відображаються у малюнковій піктограмі у формі ромба на вашій діаграмі Ганта як важливий результат для зауваження.



5. Управління ресурсами - це в основному питання про те, хто повинен що робити, використовуючи які матеріали тощо. У наведеному нижче проекті Гвен та Ділан призначені виконувати роботи з просування по службі, починаючи з 14 серпня 2018 року.



Поради щодо підготовки діаграми Ганта

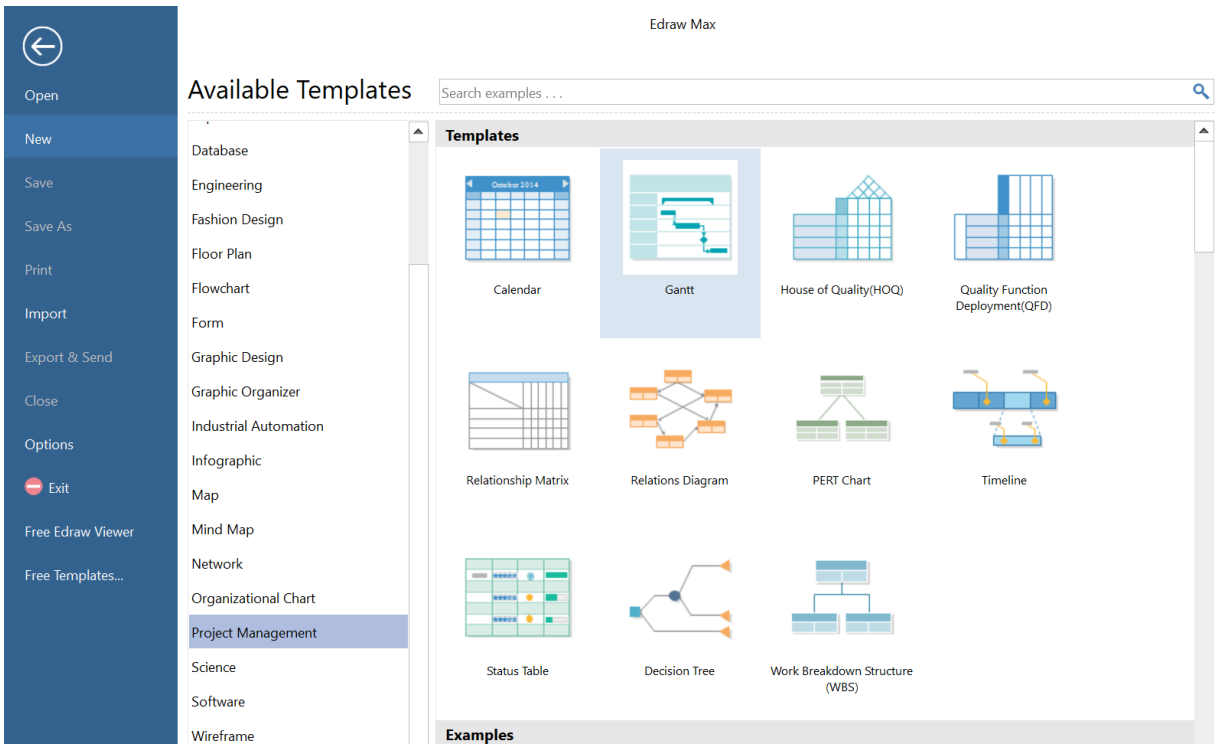
Кілька важливих порад, які ви можете мати на увазі під час підготовки діаграми Ганта, включають:

- Підготуйте список кожного завдання, яке потрібно виконати. Додайте кожне завдання в окремий рядок.
- Обов’язково перелічіть завдання в порядку їх виконання
- Визначте стандартний колір та форму брусків, які будуть використовуватися для показу прогресу. Обов’язково використовуйте той самий стиль форматування, щоб діаграма Ганта була послідовною та легко читалася
- Додайте якомога більше описів, щоб зробити діаграму інформативною. Однак не перенаселяйте занадто багато тексту
- Подумайте про використання пунктирних вертикальних сполучних ліній, щоб показати, коли завдання переходить на наступну фазу

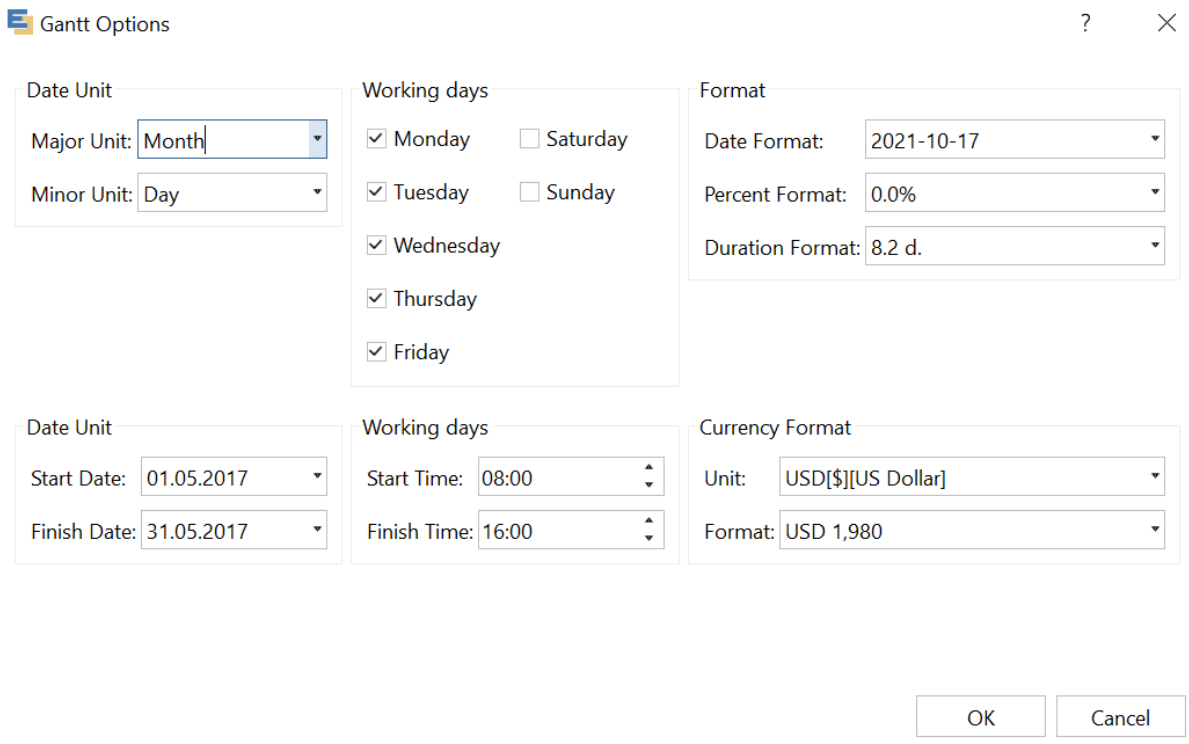
Хід виконання роботи

1. Створіть новий документ діаграми Ганта

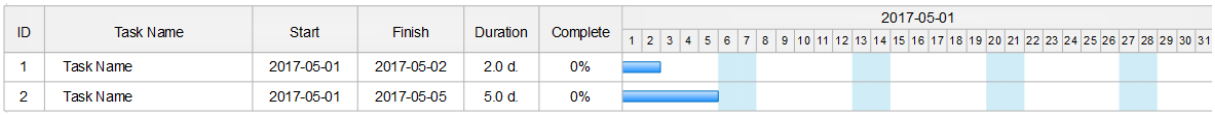
Коли Edraw запущено, виберіть відповідний вибір у меню: File > New > Project Management > Gante



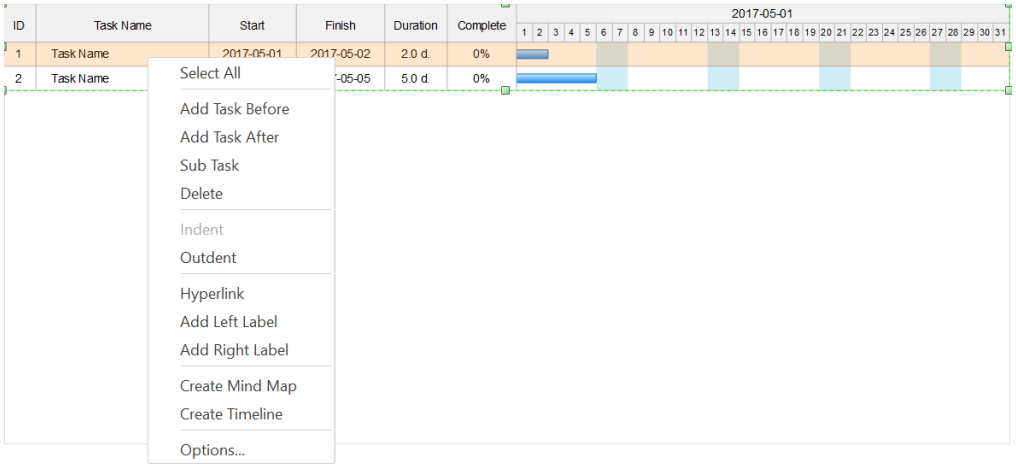
Коли відкриється новий документ діаграми Ганта, ви можете перетягнути шаблон діаграми Ганта з панелі бібліотеки. Нижче показано діалогове вікно Параметри діаграми Ганта.



Натисніть OK, щоб прийняти параметри за замовчуванням і створити діаграму Ганта, яка виглядатиме так:



Викликавши контекстне меню (натиснути правою кнопкою миші на назву задачі), можна розширити діаграму Ганта.

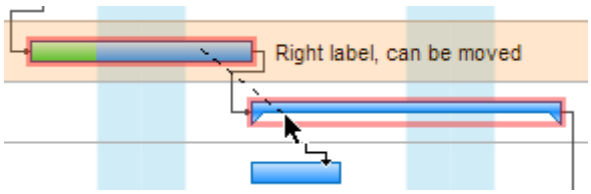


Додайте більше завдань до діаграми Ганта. Виберіть діаграму Ганта, а потім натисніть кнопку Вставити нове завдання в меню.

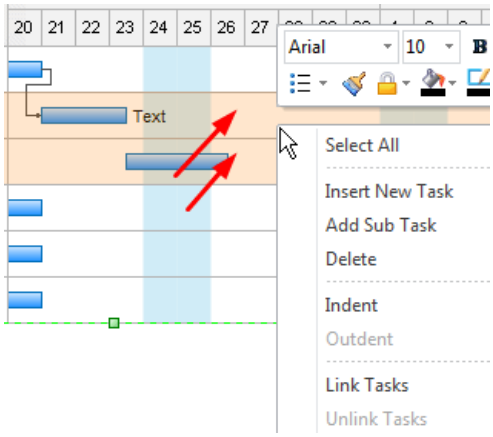
Змінити назву завдання. Двічі клацніть текст на діаграмі Ганта, а потім введіть текст.

Додайте віхи до діаграми Ганта. Ви можете змінити завдання на етап, встановивши його тривалість 0. Аналогічно, ви можете змінити етап у завдання, встановивши тривалість на позитивне число.

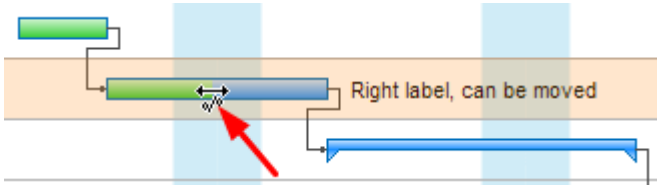
Створіть залежності між завданнями на діаграмі Ганта. Спочатку клацніть, щоб вибрати панель завдань або віху, з якої потрібно зв'язати, і перетягніть мишу до залежного завдання чи етапу.



Або натисніть клавішу Ctrl і клацніть, щоб вибрати залежне завдання або етап (Натисніть білу область в рядку). Клацніть правою кнопкою миші одну з фігур і виберіть пункт Завдання зв'язування.



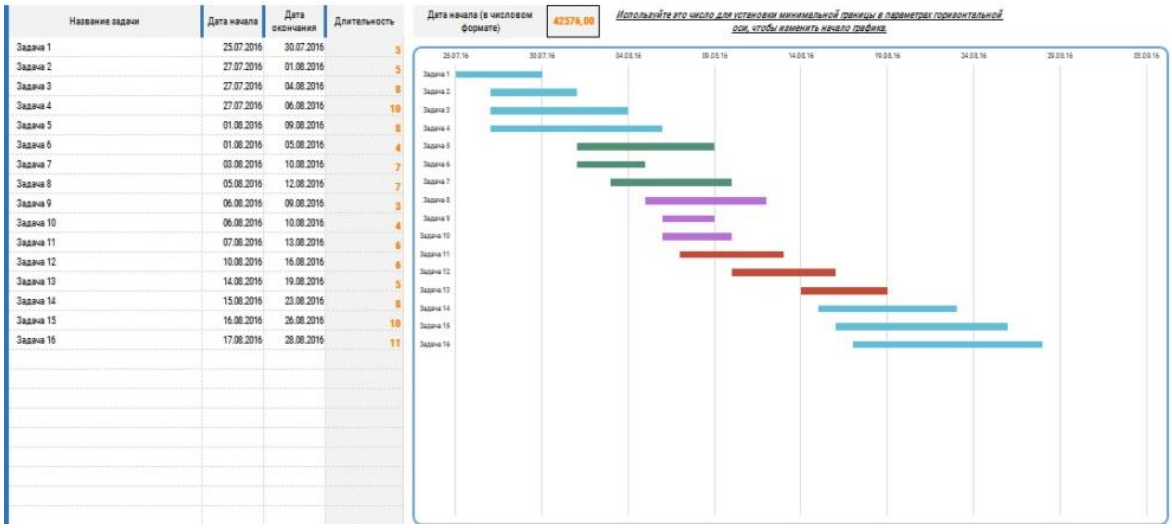
Налаштуйте панель завдань. Ви також можете змінити тривалість, вибравши панель завдань, а потім перетягнувши маркер виділення на правому кінці панелі завдань, доки він не вирівняється з новою датою завершення на часовій шкалі.



Індивідуальне завдання

Засобами Edraw Max створити діаграму Ганта для процесу заповнення медичної документації згідно свого варіанту.

Приклад діаграми Ганта



Практична робота №7

КАРТИ РОЗУМУ ДЛЯ СТРУКТУРУВАННЯ ІНФОРМАЦІЇ

Мета роботи – оволодіння прийомами створення карт розуму у програмі EdrawMax Online.

Теоретичні відомості

Коли йдеться про карти розуму, ви можете спочатку подумати про її зовнішній вигляд, навіть якщо не можете пояснити її визначення конкретно. Карта розуму - це загальна діаграма, яка використовується для наочної організації вашої інформації та концепцій. Крім того, це може допомогти вам генерувати нові ідеї та підвищити продуктивність.

Карти розуму можна малювати вручну під час зустрічі, лекції чи обговорення. Останнім часом стало популярним створювати ментальну карту з інструментами створення діаграм, оскільки вони працюють швидше і ефективніше справляються зі складними структурами (рис. 7.1).

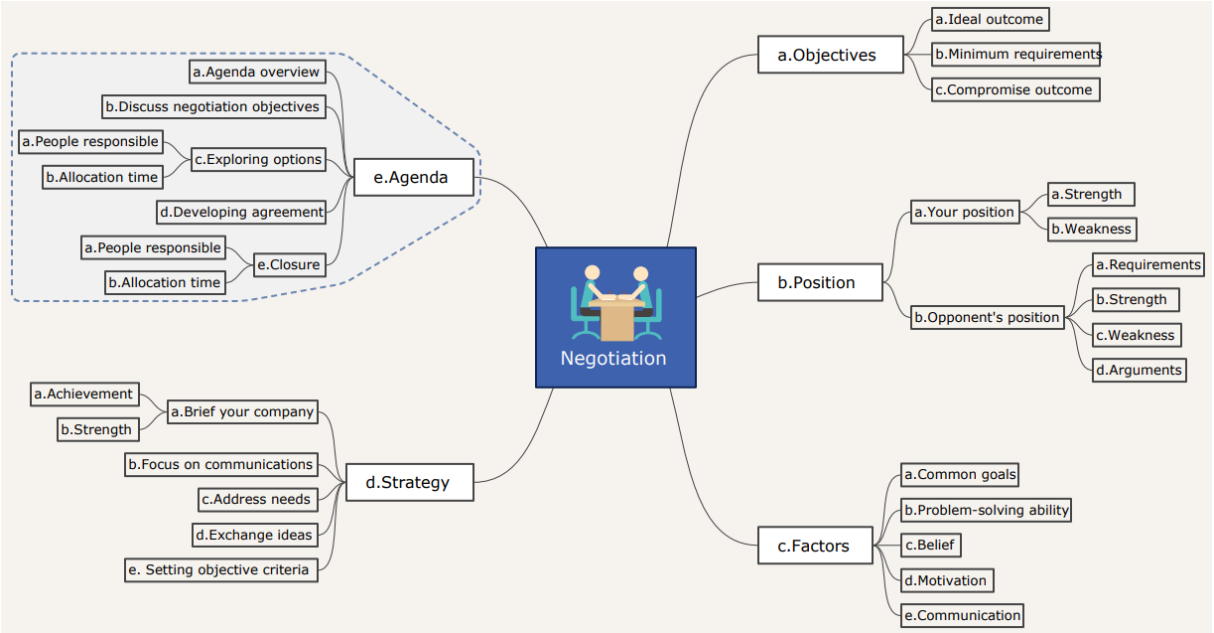


Рисунок 7.1 – Приклад карти розуму

Переваги використання карт розуму

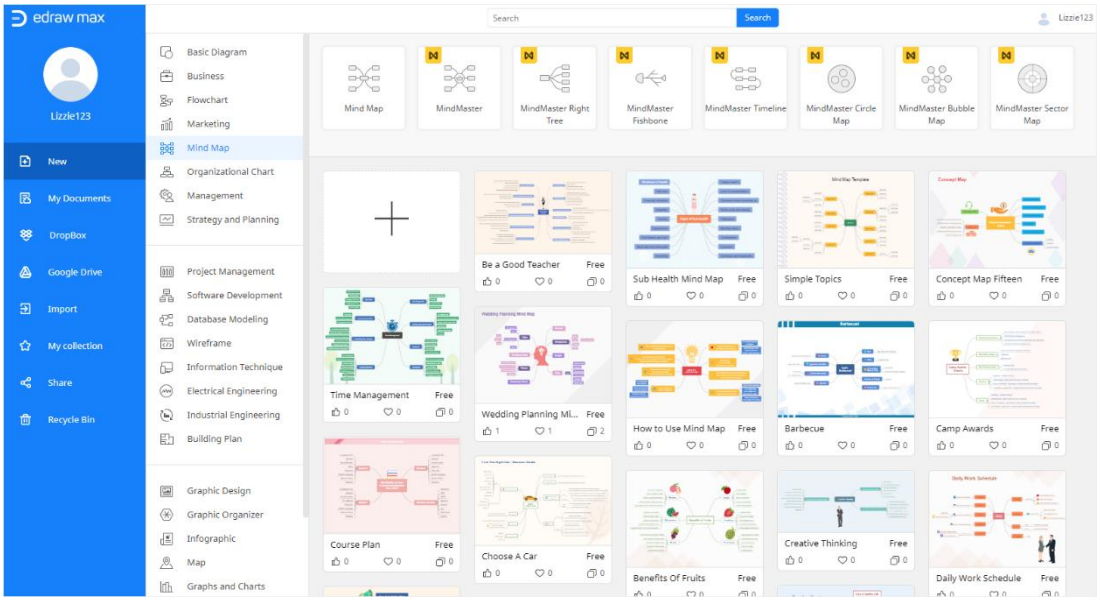
Кожен може малювати або складати карти розуму у своєму навчальному закладі або працювати з інструментами картографування розуму в Інтернеті або настільному комп’ютері. Очевидно, що переваги використання карт розуму є масштабними і значними. Карти розуму можуть:

1. Допоможіть вам організувати інформацію у невеликі та легкі для читання шматки.
2. Допоможіть вам отримати повний огляд справи.
3. Допоможіть поліпшити вашу здатність всебічно та логічно мислити.
4. Допоможе вам організувати поняття з різних предметів та підготуватися до іспитів.
5. Допоможіть вам знайти творчий спосіб робити нотатки.

Існує багато прихованих переваг, особливих для тих людей, які тривалий час користувалися картами розуму під час навчання чи роботи. Коли ви берете розумові карти у своє життя, ви можете отримати з них цінність і знайти більше інтересів.

Як створити карту розуму в Інтернеті

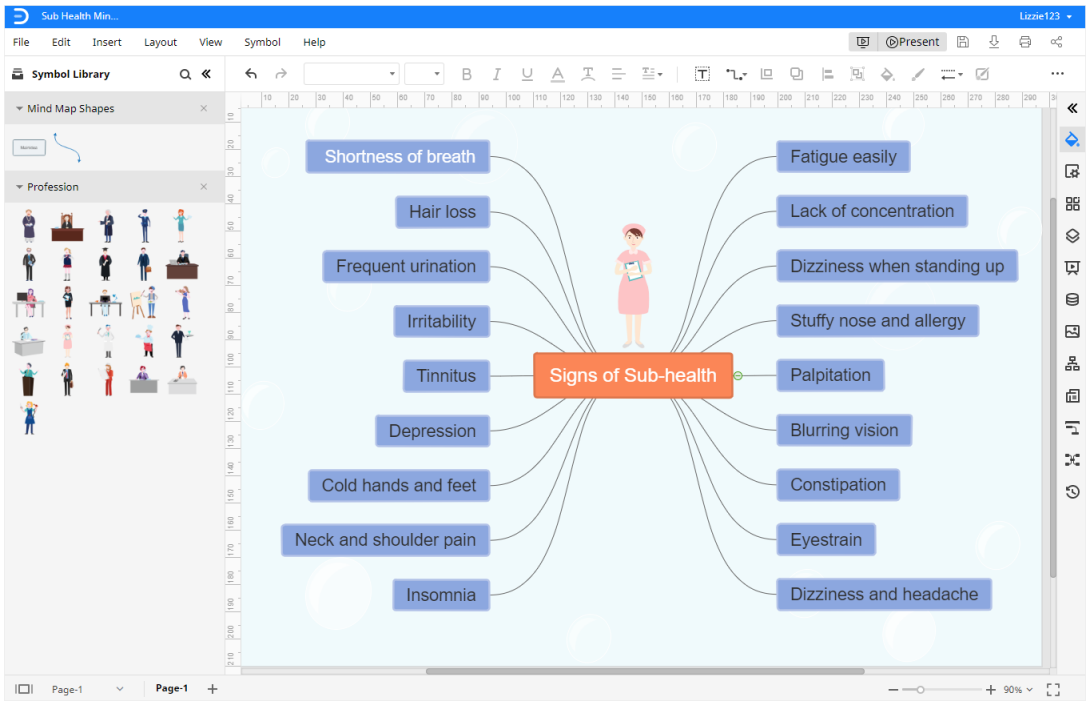
Як створити ментальну карту у розробнику ментальної карти? Це так само просто, як триступеневий процес. Хоча як створити розумну карту в Інтернеті? З правильними інструментами буде легко. Виконайте наведені нижче кроки та дізнайтесь, як створити карту розуму за допомогою потужного виробника - **EdrawMax Online** .



Крок 1 Почніть із порожньої сторінки

Увімкніть (**EdrawMax Online**) у своєму веб **-переглядачі** , і для цього можна використовувати 2 різні способи.

Щоб використовувати шаблон ментальної карти як структуру, ви можете вибрати будь -кого, хто вам подобається, у галереї шаблонів або в шаблонах Edraw. Відкривши його у програмі, ви зможете вільно налаштувати шаблон.



Щоб побудувати карту розуму з нуля, виберіть «**Карта розуму**» в області типу горизонтальної діаграми, натисніть значок плюса на галереї та

відкрийте порожній шаблон, після чого ви зможете додати основні теми та підтеми навколо основної ідеї, щоб збільшити карту розуму.

Крок 2 Додайте теми та підтеми

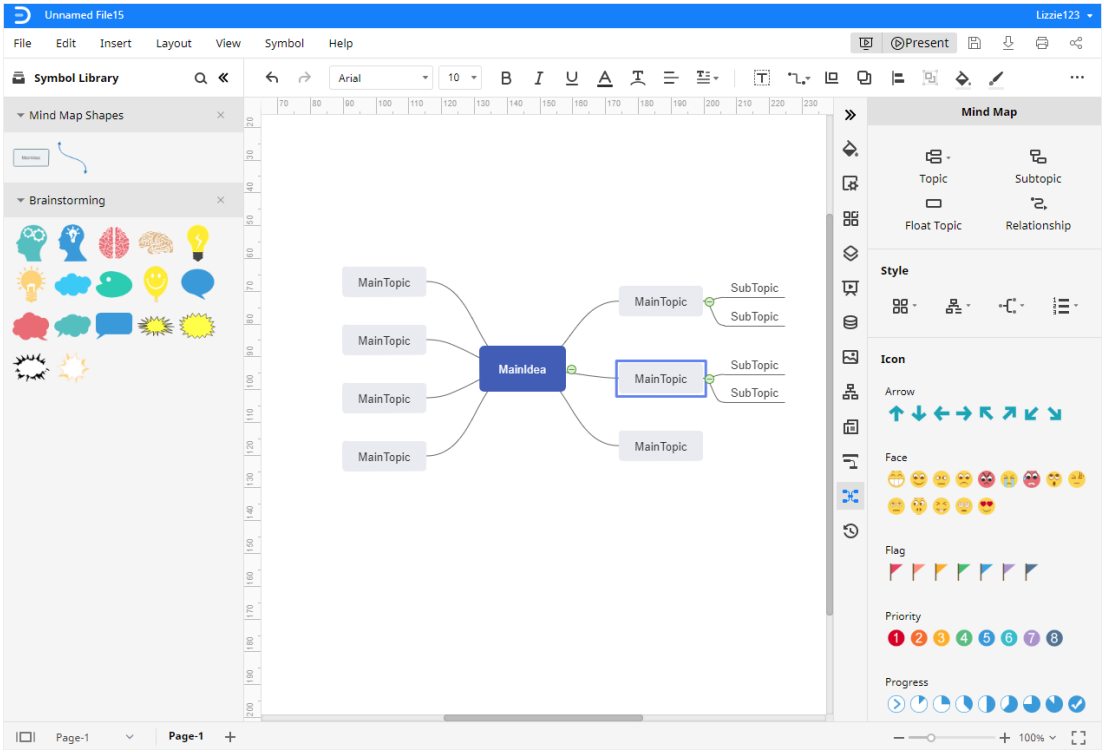
Коли шаблон порожньої карти розуму відкрито в **EdrawMax Online** , форми карти розуму відображатимуться на панелі «**Бібліотека**».

Просто перетягніть форму **Основної ідеї** (першу на **фігурах карти розуму**) і опустіть її на полотно. Виберіть **Основна ідея** , натисніть клавішу **Enter** на клавіатурі, щоб додати навколо неї форми **Основної теми**; або натисніть **Тема> Вставити тему** на панелі **Карти розуму** .

Виберіть одну з основних тем, клацніть **Підтема** на панелі, а потім натисніть клавішу **Enter,** щоб додати інші підтеми.

Усі теми будуть автоматично з'єднані в **EdrawMax Online** , тому вам не потрібно думати про те, як додати сполучні рядки на карті розуму.

Крім того, ви можете двічі клацнути на полях з темами та безпосередньо відредагувати текст у них.

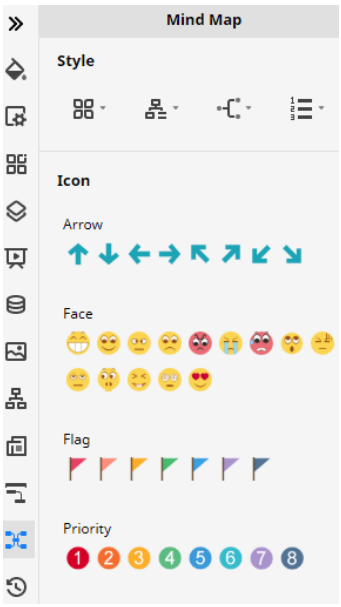


Крок 3 Повторіть той самий процес

Зрештою, ви можете повторювати той самий процес додавання тем і підтем, поки вам не вистачить кількості блоків тем. Водночас підтеми

нижчого рівня будуть створені, коли інформація, яку ви хочете представити, буде глибшою та детальнішою.

Як легко змінити стиль карти розуму



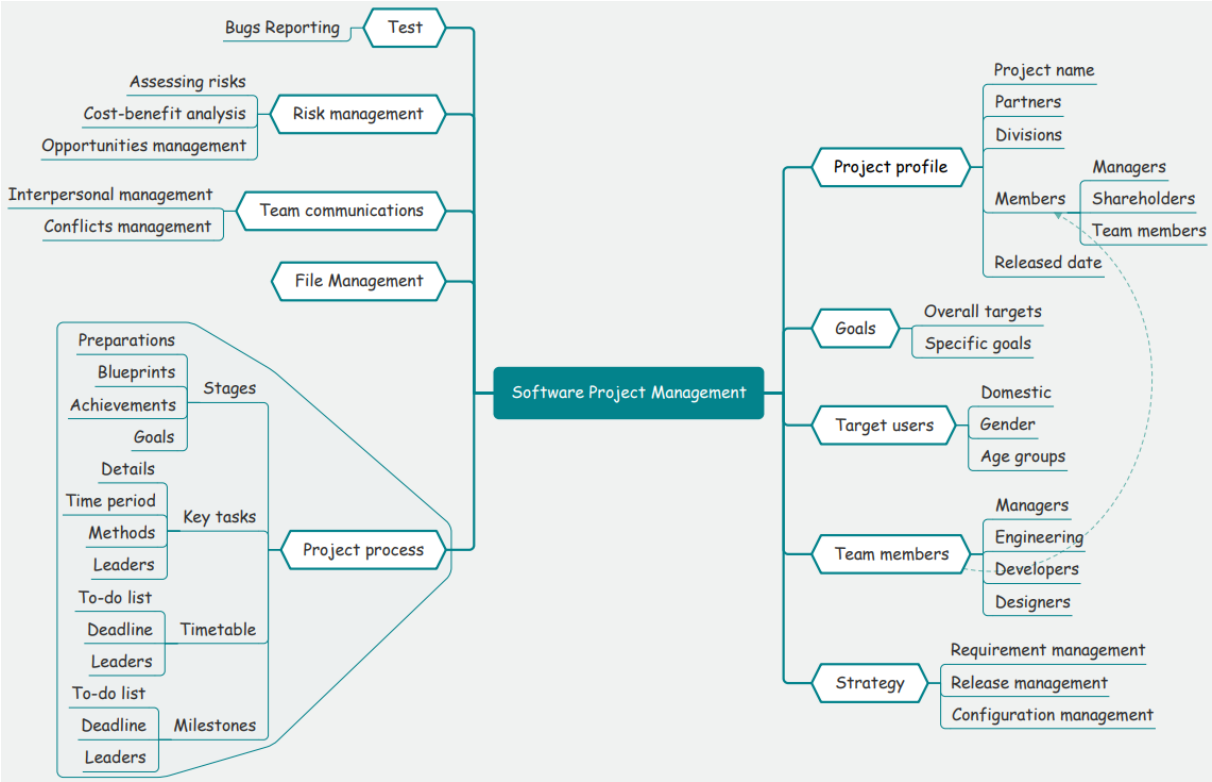
Після того, як карта розуму буде завершена, ви можете скористатися інструментами форматування на панелі **карти розуму** та змінити її тему, макет, стиль роз'єму та тип нумерації одним натисканням кнопки.

Крім того, ви також можете вставляти у теми різні іконки. Наприклад, ви можете додати значки пріоритету до тем, щоб ви могли вказати, яка з них є найважливішою або терміновою для вас.

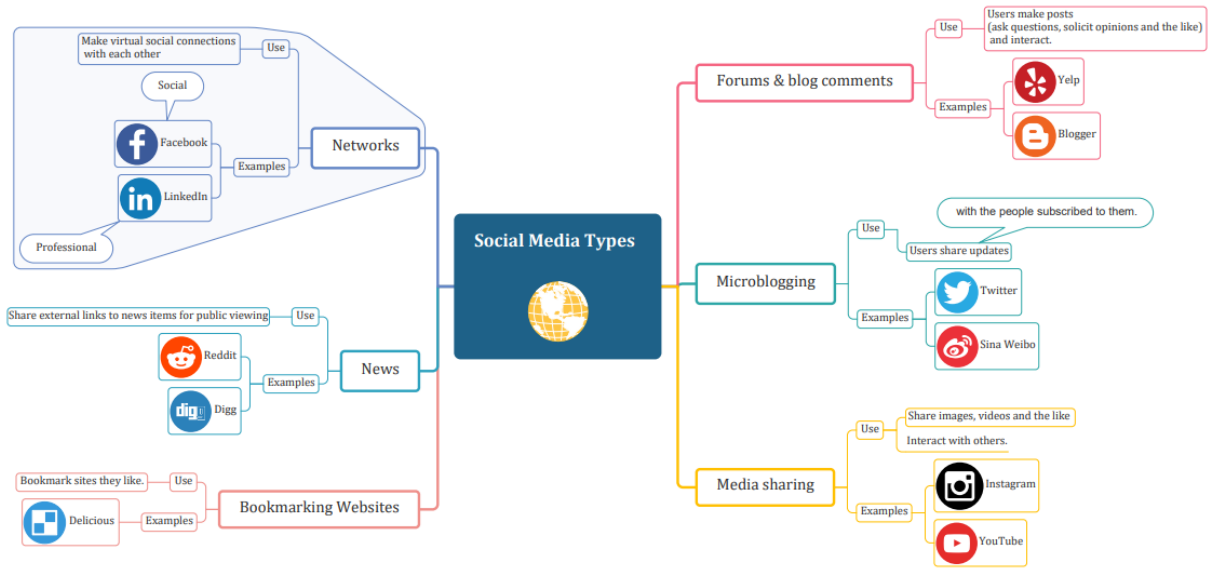
Приклади карт розуму

Якщо ви хочете створити карти розуму в різних стилях або темах, тут ми представимо вам кілька цікавих і значущих прикладів карток розуму.

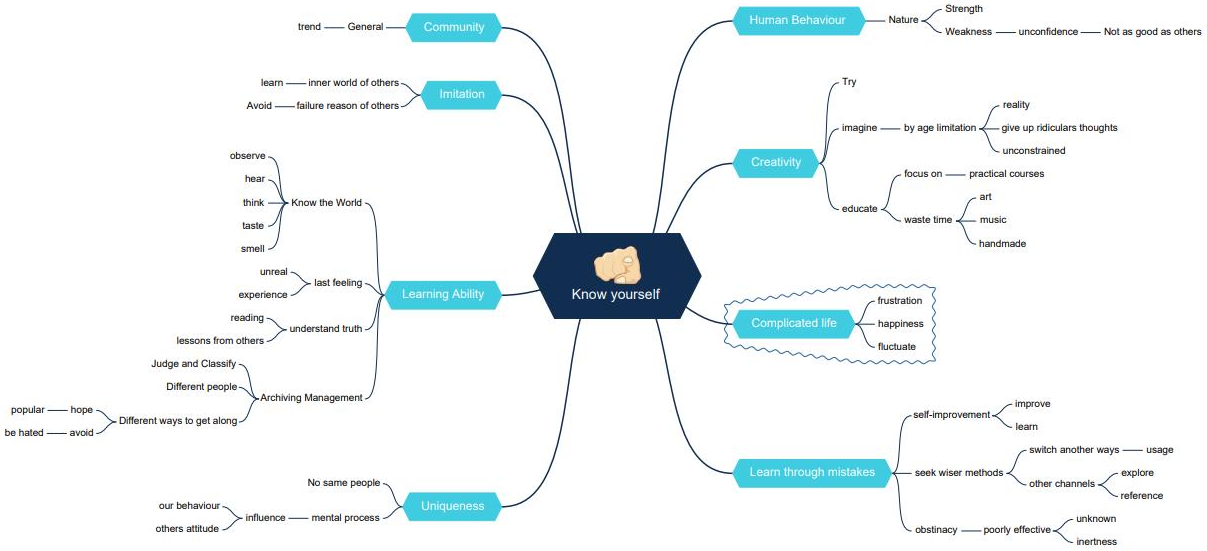
Якщо ви хочете поставити цілі на певний період свого життя, ви можете використовувати карту розуму для складання плану.



Карта розуму вище показує процес управління проектами програмного забезпечення. Тому ви можете використовувати його, коли вам потрібно скласти подібну карту розуму.



Третя карта розуму показує вам різні типи соціальних медіа, і ви можете налаштувати її відповідно до ваших вимог.



Ви побачите, що карти розуму також можуть допомогти вам пізнати та проаналізувати себе з різних аспектів.

Індивідуальне завдання

Засобами EdrawMax Online створити карту розуму по структурі медичної документації згідно свого варіанту.

Приклад створення карти розуму для роботи медичної реєстратури наведено на рисунку 7.2.

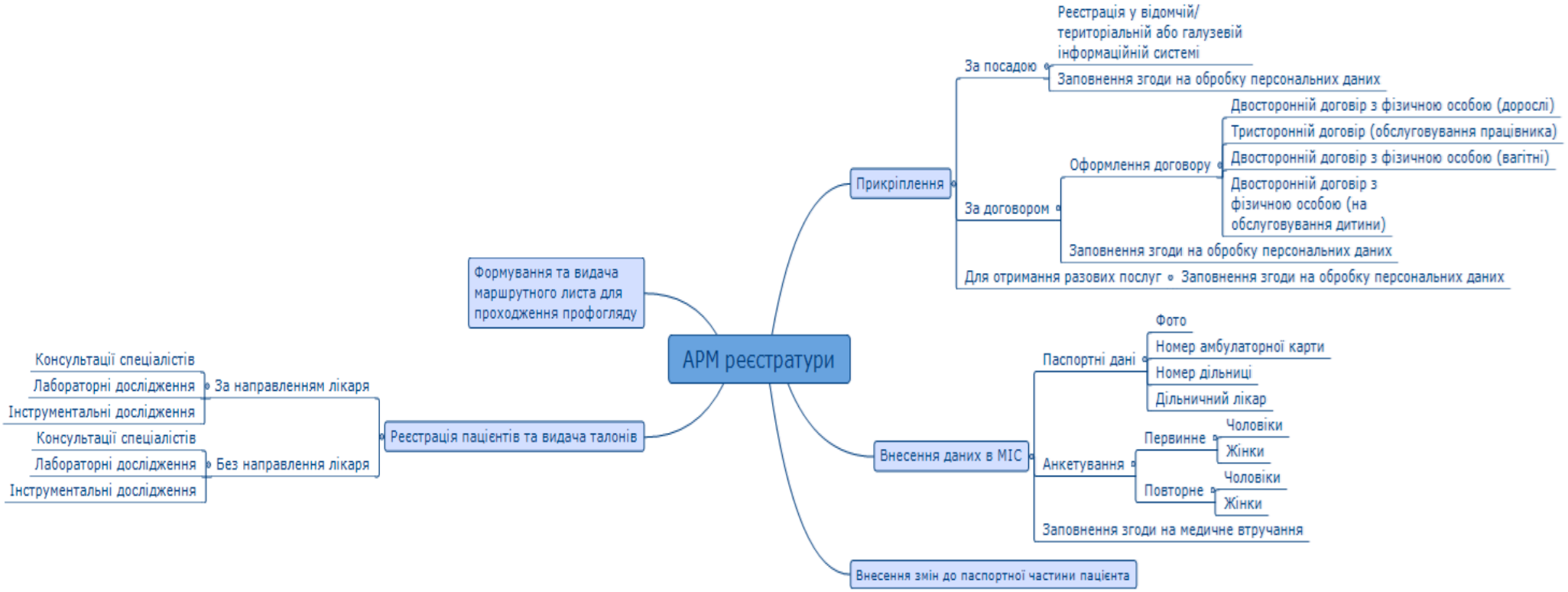


Рисунок 7.2 – Карта розуму автоматизованого робочого місця реєстратури

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

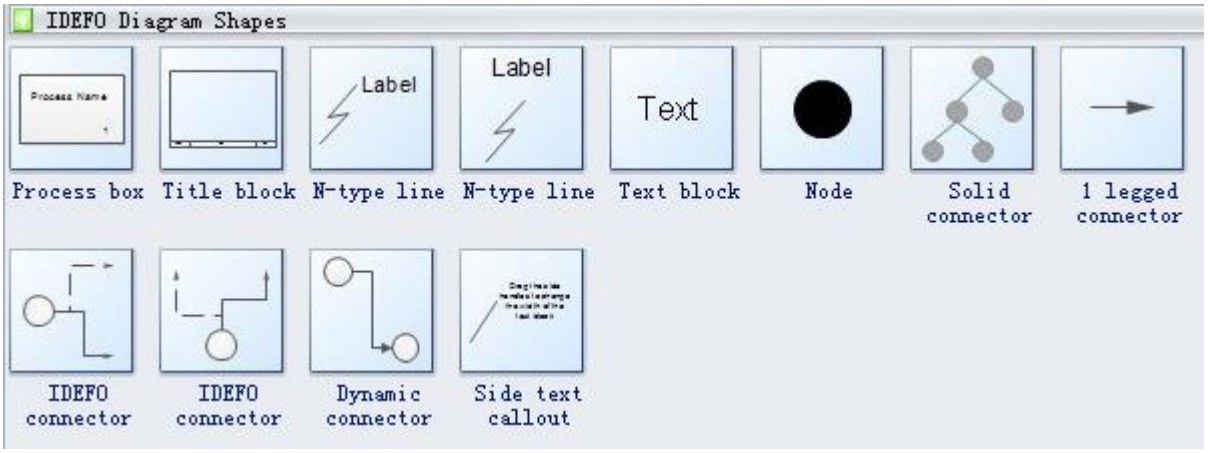
1. Проектування інформаційних систем: Загальні питання теорії проектування ІС. Конспект лекцій [Електронний ресурс]: навчальний посібник для студентів спеціальності 122 «Комп'ютерні науки» / КПІ ім. Ігоря Сікорського; уклад. О.С.Коваленко, Л. М. Добровська. – Київ: КПІ ім. Ігоря Сікорського, 2020. – 192 с. Доступ: https://ela.kpi.ua/handle/123456789/33651
2. Центр підтримки Edraw - https://www.edrawsoft.com/guide/edrawmax/
3. Интеллектуальные и информационные системы в медицине: мониторинг и поддержка принятия решений: сборник статей. - М.; Берлин: Директ-Медиа, 2016. – 529 с.
4. Основы построения автоматизированных информационных систем: Учебник / В.А. Гвоздева, И.Ю. Лаврентьева. - М.: ИД ФОРУМ: НИЦ Инфра-М, 2013. - 320 с.
5. Сабанов, В.И. Информационные системы в здравоохранении/ В.И. Сабанов, А.Н. Голубев, Е.Р. Комина.– Ростов-на Дону: Феникс, 2007 г. – 224 с.
6. Грекул В.И. Проектирование информационных систем / В.И. Грекул, Г.Н. Денищенко, Н.Л. Коровкина, 2-е изд., испр. – М.: Бином. Лаборатория знаний Интуит, 2008. – 300 с.
7. Кобринский, Б.А. Автоматизированные регистры медицинского назначения: теория и практика применения: монография / Б.А. Кобринский. - Изд. 2-е, стер. - М.; Берлин: Директ-Медиа, 2016. - 149 с.
8. Буч Г., Рамбо Д., Якобсон И. Б90 Язык UML. Руководство пользователя. 2-е изд.: Пер. с англ. Мухин Н. –. М.: ДМК Пресс, 2006.
9. Маклаков С.В. Моделирование бизнес-процессов с ALLFusion Process Modeler. 2-е изд., испр. и доп. / С.В. Маклаков – М.: Диалог-МИФИ, 2007. – 224с.
10. Мацяшек Л. Анализ требований и проектирование систем. Разработка информационных систем с использованием UML.: Пер. с англ.: / Л. Мацяшек –М.: Вильямс, 2002. – 432 с.
11. Структурный анализ систем: IDEF-технологии // С. В. Черемных, И.О.Семенов, В.С. Ручкин. – М.: "Финансы и статистика", 2003. - 208 с.

Додаток 1

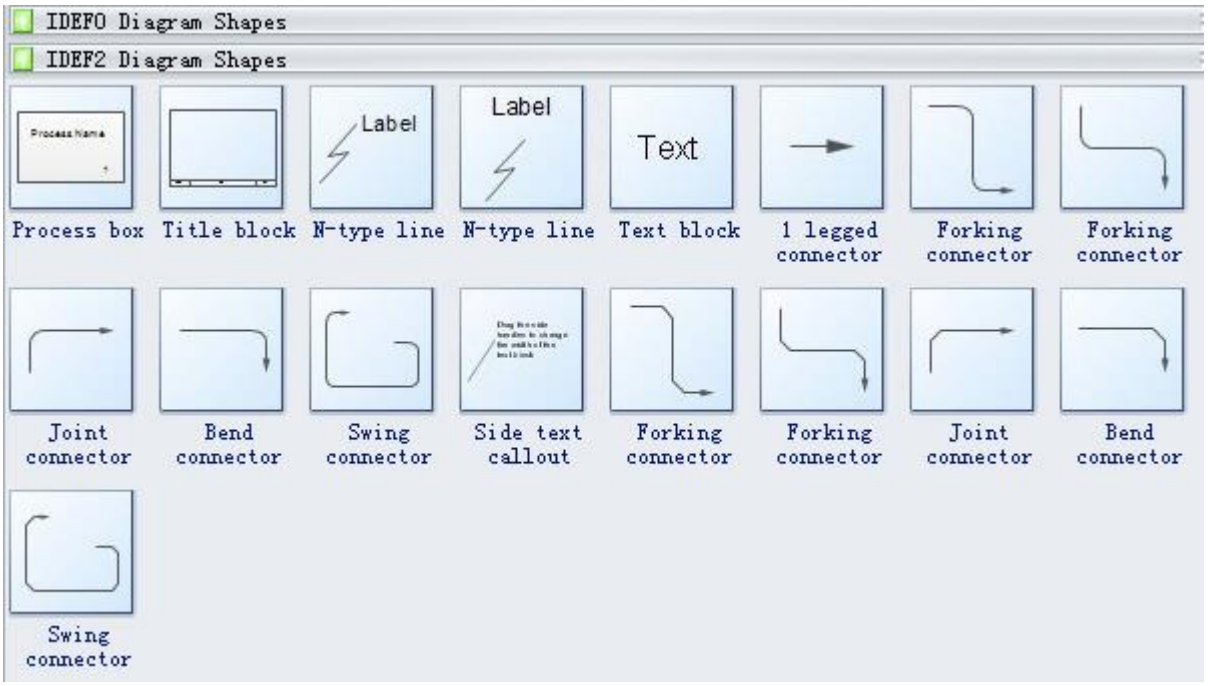
Зображення основних елементів в Edraw Max

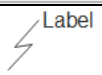
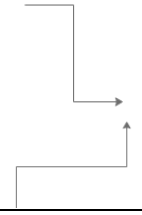
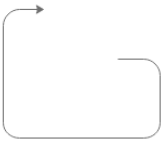
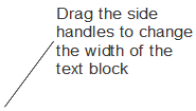
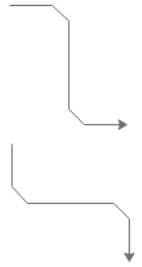
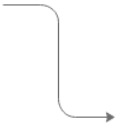
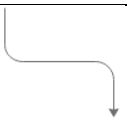
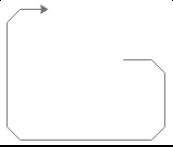
ЕЛЕМЕНТИ ДІАГРАМ СімействА IDEF

IDEF0 Diagram Shapes

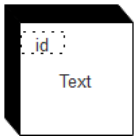
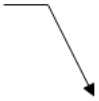
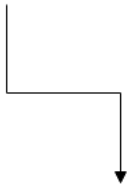
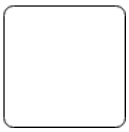
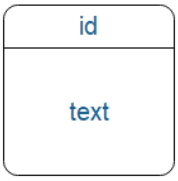
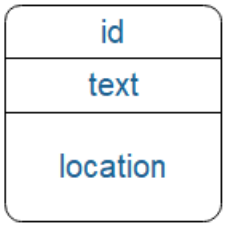
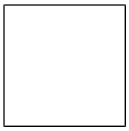
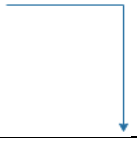


IDEF2 Diagram Shapes

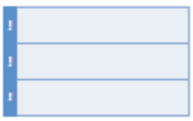


Назва	Зображення	IDEF0	IDEF2	Назва	Зображення	IDEF0	IDEF2
Process box		+	+	Node		+	-
Title block		+	+	Solid connector		+	-
N-type line		+	+	1 legged connector		+	+
Text block		+	+	IDEF0 connector		+	-
Dynamic connector		+	-	Swing connector		-	+
Side text callout		+	+	Bevel forking connector		-	+
Forking connector		-	+	Bevel joint connector		-	+
Vertical forking connector		-	+	Bevel bend connector		-	+
Joint connector		-	+	Bevel swing connector		-	+
Bend connector		-	+				

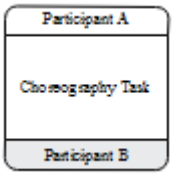
ЕЛЕМЕНТИ ДІАГРАМИ DFD

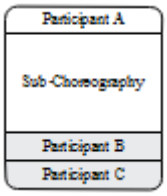
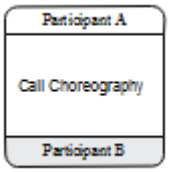
Назва елементу	Графічне зображення	Назва елементу	Графічне зображення
External Entity		Angled Connector	
Entity		Top to Bottom Variable	
Process 2		Bottom to Side	
Process		Side to Side	
Process w/location		Side to Same Side	
Data Store		Top to Top Side	
Interface		Straight Connector	
Jump		Dynamical Connector	
		Line curve connector	

ЕЛЕМЕНТИ ДІАГРАМИ BPMN

Назва елементу	Графічне зображення	Назва елементу	Графічне зображення
Задача Task		Паралельна множинна подія Parallel Multiple Event	
Шлюз Gateway		Паралельна множинна подія 2 Parallel Multiple Event 2	
Початок Start		Паралельна множинна подія 3 Parallel Multiple Event 3	
Середина Intermediate		Паралельна множинна подія 4 Parallel Multiple Event 4	
Кінець End		Назва пула Pool Title	
Стартове повідомлення Start Message		Пул (басейн) Pool	
Стартове повідомлення 2 Start Message 2		Блок пулів (басейнів) Pools Block	
Проміжне повідомлення Intermediate Message		Звернутий підпроцес Collapsed Sub-Process	

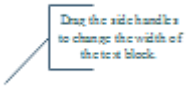
Назва елементу	Графічне зображення	Назва елементу	Графічне зображення
Проміжне повідомлення 2 Intermediate Message 2		Циклічний процес Loop Process	
Відправка повідомлення Throwing Message		Цикл паралельний – багатовіконний Loop - Parallel Multi-Instance	
Кінець повідомлення End Message		Цикл послідовний багатоекземплярний Loop - Sequential Multi-Instance	
Таймер запуску Start Timer		Компенсація Compensation	
Таймер запуску 2 Start Timer 2		Спеціальний процес Ad-Hoc Process	
Проміжний таймер Intermediate Timer		Підпроцес циклу Loop Sub-Process	
Проміжний таймер 2 Intermediate Timer 2		Множинний екземпляр Multiple Instance	
Початок ескалації Start Escalation		Підпроцес компенсації Compensation Sub-Process	
Початок ескалації 2 Start Escalation 2		Спеціальний підпроцес Ad-Hoc Sub-Process	

Назва елементу	Графічне зображення	Назва елементу	Графічне зображення
Проміжна ескалація Intermediate Escalation		Транзакція Transaction	
Проміжна ескалація 2 Intermediate Escalation 2		Підпроцес події Event Sub-Process	
Проміжна ескалація 3 Intermediate Escalation 3		Активність виклику Call Activity	
Кінець ескалації End Escalation		Розмова Conversation	
Помилка запуску Start Error		Звернута розмова Collapsed Conversation	
Проміжна помилка Intermediate Error		Розмова по телефону Call Conversation	
Кінець помилки End Error		Звернута розмова по телефону Collapsed Call Conversation	
Умови запуску Start Condition		Група Group	
Умови запуску 2 Start Condition 2		Задача по хореографії Choreography Task	

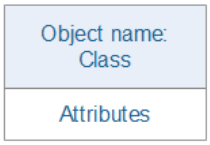
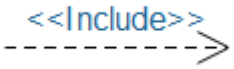
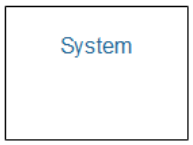
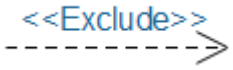
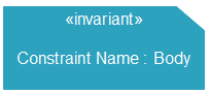
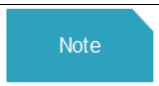
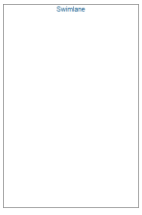
Назва елементу	Графічне зображення	Назва елементу	Графічне зображення
Проміжний стан Intermediate Condition		Суб-хореографія Sub-Choreography	
Проміжний стан 2 Intermediate Condition 2		Виклик хореографії Call Choreography	
Відміна Cancel		Об’єкт даних Data Object	
Відміна 2 Cancel 2		Об’єкт колекції даних Collection Data Object	
Початок посилання Start Link		Вхідні дані Input Data	
Проміжне посилання Intermediate Link		Вихідні дані Output Data	
Проміжне посилання 2 Intermediate Link 2		Сховище даних Data Store	
Кінець посилання End Link		Ексклюзивний шлюх з маркером Exclusive Gateway with Marker	
Компенсація Compensation		Шлюз на основі подій Event-Based Gateway	

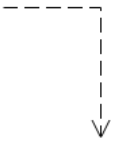
Назва елементу	Графічне зображення	Назва елементу	Графічне зображення
Компенсація 2 Compensation 2		Ексклюзивний шлюх на основі подій Exclusive Event-based Gateway	
Компенсація 3 Compensation 3		Паралельний шлюх на основі подій Parallel Event-based Gateway	
Компенсація 4 Compensation 4		Паралельний шлюз Parallel Gateway	
Сигнальна подія Signal Event		Інклюзивний шлюз Inclusive Gateway	
Сигнальна подія 2 Signal Event 2		Комплексний шлюз Complex Gateway	
Сигнальна подія 3 Signal Event 3		Послідовний потік Sequence Flow	
Сигнальна подія 4 Signal Event 4		Послідовність потоку під прямим кутом Sequence Flow Right-angle	
Сигнальна подія 5 Signal Event 5		Потік повідомлень Message Flow	
Сигнальна подія 6 Signal Event 6		Потік повідомлень під прямим кутом Message Flow Right-angle	

Назва елементу	Графічне зображення	Назва елементу	Графічне зображення
Багаторазова подія Multiple Event		Асоціація даних Data Association	
Багаторазова подія 2 Multiple Event 2		Асоціація Association	
Багаторазова подія 3 Multiple Event 3		Асоціація під прямим кутом Association Right-angle	
Багаторазова подія 4 Multiple Event 4		Маркер процесу Process Marker	
Багаторазова подія 5 Multiple Event 5		Маркер петлі Loop Marker	
Багаторазова подія 6 Multiple Event 6		Паралельний MI маркер Parallel MI Marker	
Бізнес-правило Задача Business Rule Task		Послідовний MI маркер Sequential MI Marker	
Сервісна задача Service Task		Ad-Hoc маркер Ad-Hoc Marker	
Задача сценарію Script Task		Маркер компенсації Compensation Marker	

Назва елементу	Графічне зображення	Назва елементу	Графічне зображення
Прямокутна повітряна куля Rectangle balloon		Відправити завдання Send Task	
Анотації Annotation		Отримати завдання Receive Task	
Ручна задача Manual Task		Задача користувача User Task	

ЕЛЕМЕНТИ ДІАГРАМИ Use Case

Назва елементу	Графічне зображення	Назва елементу	Графічне зображення
Actor		Straight connector	
Use Case		Communication	
Package		Generalization	
Object		Include	
System Boundary		Exclude	
Constraint		Interface	
Note			
Activity		Horizontal Synchronization Bar	
State		Vertical Synchronization Bar	
Object in State		Initial State	
Decision Activity		Final State	
Actor		Swimlane	
Control Flow		Multiple Trigger	*

Назва елементу	Графічне зображення	Назва елементу	Графічне зображення
Control Flow 2		Symbol <<and>>	« »
Object Flow		Constraint	
Object Flow 2		Note