**Практична робота №13 Прикладні та теоретичні методи програмування**

**На період** карантину в дистанційній формі навчання на надані запитання потрібно надати письмові відповіді, надіславши їх на електронну адресу викладача. Файл надавати з іменем у форматі

**PI<Номер групи><Номер лекції / практичної / лабораторної>[-<Номер завдання>][літера позначення типу роботи L – лекція, P –практична, R – лабораторна]<Прізвищеанглійською>**. Наприклад, **PI3104L**buts.doc. Відповіді на запитання повинні бути не довгими і змістовними. Не копіюйте фрагментів з різних інформаційних джерел, подумайте і викладіть свою точку зору. При наявності відповідей-"близнюків" відповідь буде зараховуватися першому за часом надсилання.

**Строк виконання цієї роботи 21.05.2020**

**Мета:** Навчиться використовувати прикладні програмування для розроблюваного програмного забезпечення на прикладі SADT. Визначити поняття структурно-функціонального аналізу. Визначити основні елементи для будування методології SADT. Навчитися будувати SADT-модель для ПЗ.

**Завдання:**

1. Опрацювати теоретичні відомості. Перевірити засвоєння Вами матеріалу на контрольних запитаннях. Прочитати методичні вказівки.
2. Для проекту, над яким працювали на попередніх лабораторних та практичних роботах, побудувати функціональну модель об’єкту. Вимоги:

* Модель повинна містити контекстну діаграму.
* Визначити мету, точку зору моделі.
* Задати входи, виходи, механізми і управління.
* Створити декомпозицію контекстної діаграми, що складається з 3-4 блоків.
* Встановити зв'язки між блоками. Задати імена дуг.
* Модель повинна включати діаграму дерева.

1. По закінченню лабораторну роботу потрібно здати на перевірку викладачеві, надіславши електронною поштою. Якщо викладач знаходить помилки чи неточності, він може повернути роботу на доопрацювання.

**Контрольні питання для самоперевірки**

1. Для чого була розроблена методологія структурного аналізу й проектування SADT?
2. Наведіть визначення терміну "Модель SADT".
3. Яку мету має модель SADT?
4. Чим є суб'єкт в моделі SADT?
5. Що представляють блоки моделі SADT?
6. Що представляють дуги моделі SADT?
7. Які типи взаємозв'язків мають дуги між блоками?

**Методичні вказівки - алгоритм аналізу системи**

*Збір інформації про систему. Вибір мети і точки зору*

Це основоположні параметри моделі. Мета вибирається на основі переліку питань, на які модель повинна відповісти. Точка зору представляє позицію, з якої описується система.

*Створення діаграми A0 і діаграми декомпозиції A0*

1. Складання списку даних. Це список об'єктів, що мають значення на даному рівні декомпозиції. Функціональна декомпозиція більш ефективна, якщо починається зі складання списку даних.

2. Складання списку функцій. Це список функцій, які оперують з даними з попереднього списку. Кілька функцій можуть мати одні й ті ж дані, в той же час одна функція може використовувати кілька різних типів даних.

3. Побудова діаграми:

- Розташувати блоки на сторінці (з урахуванням домінування),

- Намалювати основні дуги, що представляють обмеження,

- Намалювати зовнішні інтерфейсні дуги,

- Намалювати всі що залишилися дуги.

*Створення діаграми A-0*

Узагальнення діаграми A-0 призводить до отримання верхньої діаграми моделі A-0. Ця операція дуже важлива, так як дозволяє перевірити чи адекватно назва моделі того, що робить система, переконатися в повноті зовнішніх інтерфейсів системи (дуг), виконати остаточне затвердження цілі і точки зору моделі. Створення діаграми A-0 полягає в зображенні блоку A0 і записи мети і точки зору під цим блоком.

*Подальша декомпозиція (декомпозиція обмежених об'єктів)*

Діаграма першого рівня декомпозиції A0, а також всі наступні діаграми декомпозиції, надають інтерфейсні обмеження (контекст) для дочірніх діаграм. Крім того, модель вже має на меті і точкою зору. Це робить процес подальшого проектування більш формалізованим і необхідний ступінь деталізації досягається виконанням наступного рекурсивного процесу:

1. Вибір блоку діаграми. Декомпозицію рекомендується починати з самого змістовного блоку з точки зору домінування, функціональної складності і впливу на декомпозицію інших блоків цієї діаграми. Кращим для початку декомпозиції не обов'язково буде найскладніший для розуміння блок.

2. Розгляд об'єкта, визначеного цим блоком.

3. Створення нової діаграми (за алгоритмом, подібного побудови діаграми A0).

4. Виявлення недоліків нової діаграми.

5. Створення альтернативних декомпозицій.

6. Коригування нової діаграми.

7. Коригування всіх пов'язаних з нею діаграм.

*Коли зупиняти декомпозицію*

Мета моделювання містить список питань, на які повинна відповідати модель. Коли ми можемо по діаграмах, що становить модель, знайти відповіді на ці питання, мета моделювання вважається досягнутою загальні витрати на виготовлення моделі можна припиняти.

Якщо ж мета моделювання ще не досягнута, то необхідно виконувати декомпозицію тих функцій, розуміння роботи яких дасть відповіді на необхідні питання. При цьому необхідно строго дотримуватися кордону моделі, закладені в контекстної діаграмі, і діаграми декомпозиції першого рівня A0.

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

Методологія SADT (Structured Analysis and Design Technique) розроблена Дугласом Россом. На її основі розроблена, зокрема, відома методологія IDEF0 (Icam DEFinition). Методологія SADT являє собою сукупність методів, правил і процедур, призначених для побудови функціональної моделі об’єкта будь-якої предметної області. Функціональна модель SADT відображає функціональну структуру об’єкта, тобто вироблені їм дії та зв’язку між цими діями. Основні елементи цієї методології грунтуються на наступних концепціях:

* *графічне представлення блочного моделювання*. Графіка блоків і дуг SADT-діаграми відображає функцію у вигляді блоку, а інтерфейси входу / виходу представляються дугами, відповідно входять в блок і виходять з нього. Взаємодія блоків один з одним описуються за допомогою інтерфейсних дуг, виражають “обмеження”, які в свою чергу визначають, коли і яким чином функції виконуються і управляються;
* *строгість і точність*. Виконання правил SADT вимагає достатньої строгості й точності, не накладаючи в той же час надмірних обмежень на дії аналітика. Правила SADT включають:
* обмеження кількості блоків на кожному рівні декомпозиції (правило 3-6 блоків);
* зв’язність діаграм (номера блоків);
* унікальність міток і найменувань (відсутність повторюваних імен);
* синтаксичні правила для графіки (блоків і дуг);
* поділ входів та управлінь (правило визначення ролі даних).
* відділення організації від функції, тобто виключення впливу організаційної структури на функціональну модель.

Методологія SADT може використовуватися для моделювання широкого кола систем і визначення вимог і функцій, а потім для розробки системи, яка задовольняє цим вимогам і реалізує ці функції. Для вже існуючих систем SADT може бути використана для аналізу функцій, виконуваних системою, а також для вказівки механізмів, за допомогою яких вони здійснюються.

Розроблено кілька графічних мов моделювання, які отримали назви:

- Нотація IDEF0 - для документування процесів виробництва і відображення інформації про використання ресурсів на кожному з етапів проектування систем. Функціональне моделювання.

- Нотація IDEF1 - для документування інформації про виробниче оточення системи.

- Нотація IDEF2 - для документування поведінки системи в часі.

- Нотація IDEF3 - спеціально для моделювання бізнес-процесів.

Найбільш зручною мовою моделювання бізнес-процесів є IDEF0, де система представляється як сукупність взаємодіючих робіт або функцій. Така чисто функціональна орієнтація є принциповою. Функції системи аналізуються незалежно від об'єктів, якими вони оперують. Це дозволяє більш чітко змоделювати логіку і взаємодію процесів організації.

Основу методології IDEF0 складає графічна мова опису бізнес-процесів. Модель в нотації IDEF0 являє собою сукупність ієрархічно впорядкованих і взаємопов'язаних діаграм. Кожна діаграма є одиницею опису системи і розташовується на окремому аркуші.

Модель може містити чотири типи діаграм:

• контекстну діаграму (у кожній моделі може бути тільки одна контекстна діаграма);

• діаграми декомпозиції;

• діаграми дерева вузлів;

• діаграми тільки для експозиції (FEO).

Методологія IDEF-SADT являє собою сукупність методів, правил і процедур, призначених для побудови функціональної моделі системи будь-якої предметної області. Функціональна модель SADT відображає структуру процесів функціонування системи і її окремих підсистем, тобто виконувані ними дії і зв'язки між цими діями. Для цієї мети будуються спеціальні моделі, які дозволяють у наочній формі представити послідовність певних дій. Вихідними функціональними блоками будь-якої моделі IDEF0 процесу є діяльність (activity) і стрілки (arrows) - інтерфейсні дуги.

Діяльність є деяка дія або набір дій, які мають фіксовану мету і призводять до деякого кінцевого результату. Іноді діяльність називають просто процесом. Моделі IDEF0 відстежують різні види діяльності системи, їх опис та взаємодія з іншими процесами. На діаграмах діяльність або процес зображується прямокутником, який називається блоком. Стрілка служить для позначення деякого носія або впливу, які забезпечують перенесення даних або об'єктів від однієї діяльності до іншої. Стрілки також необхідні для опису того, що саме виробляє діяльність і які ресурси вона споживає. Це так звані ролі стрілок - ICOM - скорочення перших букв від назв відповідних стрілок IDEF0. При цьому розрізняють стрілки чотирьох видів (Рис. 1):

- I (Input) - вхід, тобто все, що надходить в процес або споживається процесом.

- С (Control) - управління або обмеження на виконання операцій процесу.

- О (Output) - вихід або результат процесу.

- М (Mechanism) - механізм, який використовується для виконання процесу.

**Склад функціональної моделі**

Результатом застосування методології SADT є модель, яка складається з діаграм, фрагментів текстів і глосарію, що мають посилання один на одного. Діаграми – головні компоненти моделі, всі функції ІС і інтерфейси на них представлені як блоки і дуги. Місце з’єднання дуги з блоком визначає тип інтерфейсу. Керуюча інформація входить в блок зверху, у той час як інформація, яка піддається обробці, показана з лівого боку блоку, а результати виходу показані з правого боку. Механізм (людина або автоматизована система), який здійснює операцію, представляється дугою, що входить в блок знизу (рис.1.).

Однією з найбільш важливих особливостей методології SADT є поступове введення все більших рівнів деталізації у міру створення діаграм, що відображають модель. Декомпозиція є умовним прийомом, що дозволяє уявити систему у вигляді, зручному для сприйняття, і оцінити її складність. В результаті декомпозиції підсистеми за певними ознаками виділяються окремі структурні елементи та зв'язки між ними. Декомпозиція служить засобом, що дозволяє уникнути труднощів у розумінні системи. Глибина декомпозиції визначається складністю і розмірністю системи, а також цілями моделювання

****

Рис 1. Функціональний блок та інтерфейсні дуги

На рис. 2, де наведено чотири діаграми та їх взаємозв’язку, показана структура SADT-моделі. Кожен компонент моделі може бути декомпозований на інший діаграмі. Кожна діаграма ілюструє “внутрішнє будову “блоку на батьківській діаграмі.



Рис.2. Структура SADT-моделі. Декомпозиція діаграм

**Ієрархія діаграм**

Побудова SADT-моделі починається з представлення всієї системи у вигляді найпростішої компоненти – одного блоку і дуг, що зображують інтерфейси з функціями поза системою. Оскільки єдиний блок представляє всю систему як єдине ціле, ім’я, вказане в блоці, є спільним. Це вірно і для інтерфейсних дуг – вони також представляють повний набір зовнішніх інтерфейсів системи в цілому.

Потім блок, який представляє систему в якості єдиного модуля, деталізується на іншій діаграмі за допомогою декількох блоків, з’єднаних інтерфейсними дугами. Ці блоки представляють основні підфункції вихідної функції. Дана декомпозиція виявляє повний набір підфункції, кожна з яких представлена як блок, межі якого визначені інтерфейсними дугами. Кожна з цих підфункцій може бути декомпозирована подібним чином для більш детального уявлення.

У всіх випадках кожна підфункції може містити тільки ті елементи, які входять у вихідну функцію. Крім того, модель не може опустити будь-які елементи, тобто, як уже зазначалося, батьківський блок і його інтерфейси забезпечують контекст. До нього не можна нічого додати, і з нього не може бути нічого видалено.

Модель SADT являє собою серію діаграм із супровідною документацією, розбивають складний об’єкт на складові частини, які представлені у вигляді блоків. Деталі кожного з основних блоків показані у вигляді блоків на інших діаграмах. Кожна детальна діаграма є декомпозицією блоку з більш загальної діаграми. На кожному кроці декомпозиції більш загальна діаграма називається батьківського для більш детальної діаграми.

Дуги, що входять в блок і виходять з нього на діаграмі верхнього рівня, є точно тими ж самими, що й дуги, що входять в діаграму нижнього рівня і виходять з неї, тому що блок і діаграма представляють одну і ту ж частину системи.

На рис. 3 – 5 представлені різні варіанти виконання функцій і з’єднання дуг з блоками.

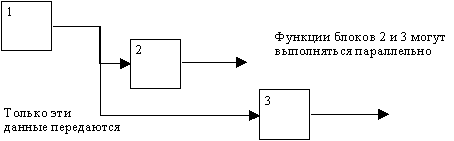
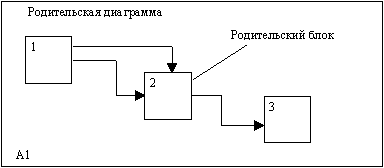


Рис. 3. Одночасне виконання



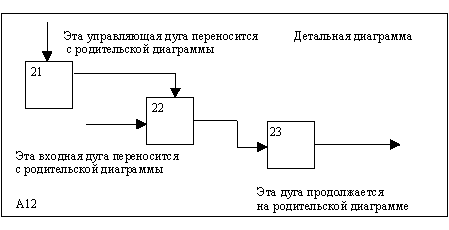


Рис. 4. Відповідність має бути повною та несуперечливою

Деякі дуги приєднані до блоків діаграми обома кінцями, в інших же один кінець залишається неприєднання. Неприєднання дуги відповідають входам, управлінням і виходів батьківського блоку. Джерело або одержувач цих прикордонних дуг може бути виявлений лише на батьківській діаграмі. Неприєднання кінці повинні відповідати дуг на вихідної діаграмі. Всі граничні дуги повинні продовжуватися на батьківській діаграмі, щоб вона була повною і несуперечливою.

На SADT-діаграмах не вказані явно ні послідовність, ні час. Зворотні зв’язки, ітерації, що тривають процеси і перекриваються (за часом) функції можуть бути зображені за допомогою дуг. Зворотні зв’язку можуть виступати у вигляді коментарів, зауважень, виправлень і т.д. (Рис. 5).

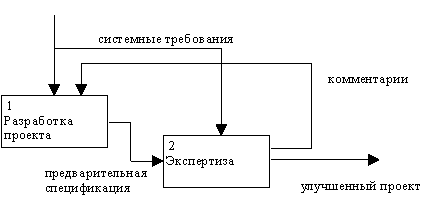


Рис. 5. Приклад зворотного зв’язку

Механізми (дуги з нижнього боку) показують засоби, за допомогою яких здійснюється виконання функцій. Механізм може бути людиною, комп’ютером або будь-яким іншим пристроєм, який допомагає виконувати цю функцію (Рис. 6).

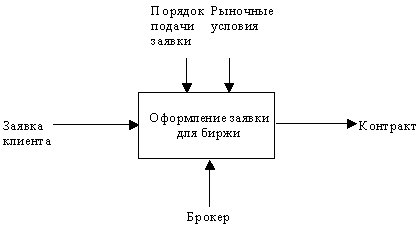


Рис. 6. Приклад механізму

Кожен блок на діаграмі має свій номер. Блок будь діаграми може бути далі описаний діаграмою нижнього рівня, яка, в свою чергу, може бути далі деталізована за допомогою необхідного числа діаграм. Таким чином, формується ієрархія діаграм. Для того, щоб вказати положення будь діаграми або блоку в ієрархії, використовуються номери діаграм. Наприклад, А21 є діаграмою, яка деталізує блок 1 на діаграмі А2. Аналогічно, А2 деталізує блок 2 на діаграмі А0, яка є самою верхній діаграмою моделі. На рис. 7 показано типове дерево діаграм.

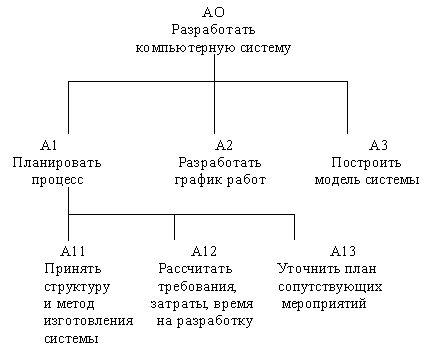


Рис. 7. Ієрархія діаграм

**Типи зв’язків між функціями**

Одним з важливих моментів при проектуванні інформаційних систем за допомогою методології SADT є точна узгодженість типів зв’язків між функціями. Розрізняють принаймні сім типів зв’язування:

|  |  |
| --- | --- |
| **Тип зв’язку** | **Відносна значимість** |
| Випадкова | 0 |
| Логічна | 1 |
| Тимчасова | 2 |
| Процедурна | 3 |
| Комунікаційна | 4 |
| Послідовна | 5 |
| Функціональна | 6 |

Нижче кожен тип зв’язку коротко визначений і проілюстровано за допомогою типового прикладу з SADT.

**(0) Тип випадкової зв’язності**: Найменш бажаний.

Випадкова зв’язність виникає, коли конкретна зв’язок між функціями мала або повністю відсутній. Це відноситься до ситуації, коли імена даних на SADT-дугах в одній діаграмі мають малий зв’язок друг з одним. Крайній варіант цього випадку показаний на рис. 8.

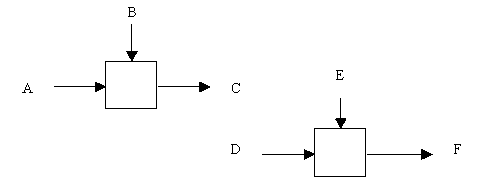


Рис. 8. Випадкова зв’язність

**(1) Тип логічної зв’язності.** Логічне зв’язування відбувається тоді, коли дані і функції збираються разом внаслідок того, що вони потрапляють у загальний клас або набір елементів, але необхідних функціональних відносин між ними не виявляється.

**(2) Тип тимчасової зв’язності.** Пов’язані за часом елементи виникають внаслідок того, що вони представляють функції, пов’язані в часі, коли дані використовуються одночасно або функції включаються паралельно, а не послідовно.



Рис.9 Тимчасова зв'язність

**(3) Тип процедурної зв’язності.** Процедурно-пов’язані елементи з’являються згрупованими разом внаслідок того, що вони виконуються протягом однієї і тієї ж частини циклу або процесу. Приклад процедурно-зв’язаної діаграми наведено на рис.º10.



Рис. 10. Процедурна зв’язність

**(4) Тип комунікаційної зв’язності.** Діаграми демонструють комунікаційні зв’язки, коли блоки групуються внаслідок того, що вони використовують одні й ті ж вхідні дані та / або виробляють одні й ті ж вихідні дані (рис. 11).

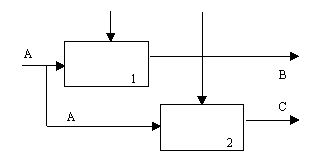


Рис.11. Комунікаційна зв’язність

**(5) Тип послідовної зв’язності.** На діаграмах, мають послідовні зв’язку, вихід однієї функції служить вхідними даними для наступної функції. Зв’язок між елементами на діаграмі є більш тісної, ніж на розглянутих вище рівнях зв’язок, оскільки моделюються причинно-наслідкові залежності (рис. 12).

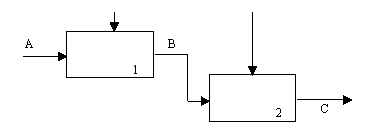


Рис. 12. Послідовна зв’язність

**(6) Тип функціональної зв’язності.** Діаграма відображає повну функціональну зв’язність, за наявності повної залежності однієї функції від іншої. Діаграма, яка є чисто функціональної, не містить чужорідних елементів, що відносяться до послідовного або слабшому типу зв’язності. Одним із способів визначення функціонально-пов’язаних діаграм є розгляд двох блоків, пов’язаних через управляючі дуги, як показано на рис. 13.

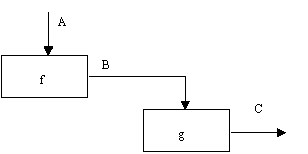


Рис. 13. Функціональна зв’язність

У математичних термінах необхідна умова для найпростішого типу функціональної зв’язності, показаної на рис. 13, має наступний вигляд:

C = g(B) = g(f(A))

Нижче в таблиці представлені всі типи зв’язків, розглянуті вище. Важливо відзначити, що рівні 4-6 встановлюють типи зв’язності, які розробники вважають найважливішими для одержання діаграм хорошої якості.

|  |  |  |  |
| --- | --- | --- | --- |
| **Значимість** | **Тип зв’язності** | **Для функцій** | **Для даних** |
| **0** | Випадкова | Випадкова | Випадкова |
| **1** | Логічна | Функції одного і того ж безлічі або типу (наприклад, “редагувати всі входи”) | Дані одного і того ж безлічі або типу |
| **2** | Тимчасова | Функції одного і того ж періоду часу (наприклад,  “Операції ініціалізації”) | Дані, що використовуються в будь-якому часовому інтервалі |
| **3** | Процедурна | Функції, які працюють в одній і тій же фазі або ітерації (наприклад, “перший прохід компілятора”) | Дані, що використовуються під час однієї і тієї ж фази або ітерації |
| **4** | Комунікаційна | Функції, що використовують одні і ті ж дані | Дані, на які впливає одна і та ж діяльність |
| **5** | Послідовна | Функції, які виконують послідовні перетворення одних і тих же даних | Дані, перетворені послідовними функціями |
| **6** | Функціональна | Функції, що об’єднуються для виконання однієї функції | Дані, пов’язані з однією функцією |