Через поступове зростання складності програмних систем, а від так і їх розробки, ще у 80-х роках минулого століття почали з’являтися спеціальні методи і засоби проектування програмного забезпечення, так звані CASE (від англ. Computer Aided Software Engineering). Сучасні CASE-засоби, на відміну від початкових, які обмежувалися лише завданнями автоматизації розробки програмного забезпечення, охоплюють широкі сфери технологій проектування інформаційних систем: від простих засобів аналізу і документування до повномасштабних засобів автоматизації, що охоплюють увесь життєвий цикл програмного забезпечення.

Найбільш трудомісткими етапами розробки інформаційних систем е етапи аналізу і проектування, у процесі яких CASE-засоби забезпечують якість прийнятих технічних рішень та підготовку проектної документації. При цьому велику роль відіграють методи візуального представления інформації. Це передбачае побудову структурних чи інших діаграм, використання різноманітної кольорової палітри, наскрізну перевірку синтаксичних правил. Графічні засоби моделювання предметної галузі дозволяють розробникам наочно бачити та вивчати існуючу інформаційну систему, перебудовувати її відповідно до поставлених цілей і наявних обмежень.

Метою статті е дослідження можливостей засобів візуального проектування, а також розгляд основних етапів проектування программного забезпечення на їх основі.

Нині на ринку інформаційних технологій налічується низка програмних продуктів CASE категорії, які підтримують існуючі підходи до розробки програмного забезпечення. До підходів проектування програмного забезпечення можна віднести такі:

- структурний, в основу якого покладено принцип алгоритмічної декомпозиції
  – структура системи описується термінами ієрархії ії функцій і передачі інформації між окремими функціональними елементами (модулями);
- об’єктно-орієнтований, який використовує об’єктну декомпозицію структура системи визначається множиною об’єктів і зв’язків між ними, а поведінка системи описується термінами обміну повідомленнями між об'єктами.

Структурний підхід мае більш тривалу історію в порівнянні з об'єктно орієнтованим, тому CASE-засоби спочатку підтримували методи структурного підходу, наприклад методологію функціонального моделювання і графічного опису процесів (IDEFO, IDEF1, IDEF3), побудову діаграм потоків даних (DFD). Найпопулярнішими у користувачів CASE-

засобами можна назвати такі інформаційні системи, як All Fusion Process Modeller, ARIS, Ramus та інші.

Завдяки активізації розвитку об’єктно-оріентованого підходу з'являється все більше програмних засобів, що реалізують методи об'єктно-оріснтованого опису інформаційних систем, наприклад Rational Rose, Visual Paradigm UML, UModel, Umbrello, Enterprise Architect та багато інших. До того ж, підтримку методів об'єктно-оріентованого підходу забезпечують і нові версії деяких CASE-систем, що традиційно вважалися структурними, зокрема система ARIS [1].

Основним методом для опису інформаційної системи відповідно до об’єктно-оріснтованого підходу е універсальна мова моделювання UML (від англ. Unified Modeling Language), розроблена в 90-х рр. Г. Бучем, Д. Рамбо та І. Якобсоном.

У сучасній версії мови UML виділяють близько 15 видів діаграм для опису структури і поведінки проектованого програмного забезпечення [2]. Однак через різноманіття видів UML-діаграм вважаемо за доцільне визначити типову послідовність дій щодо застосування цього інструменту на етапі проектування програмного забезпечення.

Етапи проектування програмного забезпечення

Перше, що потрібно зробити перед проектуванням — це *визначити користувачів* майбутньої системи і, оріснтуючись на їхні побажання, сформулювати основні вимоги до програмного продукту. Тобто спочатку потрібно дізнатися, які дії користувачі хочуть автоматизувати (перекласти на систему). Потім, поступово обмежуючи або, навпаки, розширюючи, коло функціональності і користувачів можна уточнювати межі системи. Отже, знаючи зовнішніх, по відношенню до системи, виконавців і вимоги до неї, ми можемо отримати більш чітку межу проекту.

Доступна користувачам функціональність проекту зазвичай описується в документі з назвою *«Опис* вилоз»[3]. Це потрібно для того, щоб між клієнтами і розробниками було досягнуто розуміння з питания про те, що система повинна робити і чого не повинна.

Підготувавши всю цю документацію, можна розпочинати наступний етап — проектування діаграм *прецедентів.* Кожен прецедент мас власне завдання, яке він повинен виконати, а їх набір встановлюс всі можливі шляхи використання системи.

Особливу увагу на цьому етапі слід приділяти текстовому опису, в якому вказується реакція системи на певні фактори середовища (що станеться за певних

Наступним етапом проектування програмного забезпечення с визначення його архітектури та основних складових. Цей етап проектування може забезпечити UML-діаграма *компонентів.*

Далі відповідно до об'ектно-оріснтованого підходу, необхідно виділити типи розглядуваних сутностей, їх характеристики (атрибути) і операції, які над ними можуть виконуватись (методи). В UML це можна зробити за допомогою побудови діаграми *класів,* на якій можна показати не лише типи об’єктів у вигляді класів, а й різні зв'язки між ними.

На наступному етапі проектування інформаційної системи слід визначити життєвий цикл кожного з перелічених раніше класів, тобто скласти перелік станів і переходів між ними. Для цього в UML передбачено побудову діаграм *станів,* які грунтуються на використанні апарату дискретної логіки і кінцевих автоматів.

Після визначення станів життєвого циклу всіх наявних класів доцільно відобразити взаємодії між їх об’єктами, які знаходяться в різних станах. Тут допомогти може побудова UML-діаграми *послібовностей,* яка надае можливість показати не лише об'єкти різних класів в різних станах, але й обмін повідомленнями між ними.

Однак крім технічних аспектів проектування і реалізації програмного забезпечення варто також описати технологію роботи з ним, тобто процеси розробки та експлуатації. З цією метою в UML можна скористатись засобами діаграми *діяльності,* яка дозволяс описати логічну послідовність дій та їх виконавців, а також показати пов'язані з процесами об’єкти. Отже, UML-діаграми діяльності детально відображають алгоритми виконання процесів.

Огляд сучасних CASE-засобів. У проведеному нами дослідженні проаналізовано низку засобів проектування інформаційних систем (Таблиця 1). До аналізу було обрано наступні характеристики систем: платформа, на якій може працювати система; наявність відкритого коду; чи наявна можливість генерування коду системи мови якими можна генерувати такий код; чи забезпечується функціонал оберненого інжинірингу, і якщо так, то якою мово; чи наявна підтримка стандарту обміну мѐта-інформацією (XMI).

На основі проведеного аналізу [4-12] можна виокремити найбільш вдалу і широко вживану на сьогодні систему Visual Paradigm for UML. Це система управління вимогами, яка підтримує повний цикл розробки програмного продукту — аналіз, дизайн архітектури, розробку програмного коду, тестування і розміщення продукту на стороні замовника. Visual Paradigm також забезпечує підтримку версійності і одночасної роботи команди

користувачів над одним проектом, зокрема дозволяс різним членам команди працювати над однією діаграмою і порівнювати зміни, внесені користувачами на різних етапах проекту.

Моделювання діаграм в системі реалізовано на досить високому інтуїтивно зрозумілому рівні.

Таблиця 1

Порівняння CASE-засобів проектування

Назва	Виробник	Платформа	Відкри-тий код	Мова генерування коду	Мова оберненого інжинірингу	Підтримка XMI
Erwin	Erwin	Windows	ні	-	-	ні
MS Visio	Microsoft	Windows	ні	-	-	ні
Dia	Alexander Larsson/GNOME Office	GTK+ (cross-platform)	так	-	-	ні
Enterprise Architect	Sparx Systems	Windows	ні	C++, Java, C#, VB, VB.Net, Visual Basic, Delphi, PHP, Python	C++, Java, C#, VB, VB.Net, Visual Basic, Delphi, PHP, Python	так (XMI 1.0, 1.1, 1.2)
Umbrello	Umbrello Team	Unix, Linux	так	C++, Java, C#, PHP, JavaScript, ActionScript, SQL, Pascal, Ada, Python, IDL, XML Schema, Perl, Ruby	C++, IDL, Pascal/Delphi, Ada, Python, Java, Perl	так
ArgoUML	tigris.org	Java (cross-platform	так	-	-	так (XMI 1.1, 1.2)
UModel	Altova	Windows	ні	Java 1.4, Java 5.0, Java 6.0, C# 1.2, C# 2.0, C# 3.0, VB 7.1, VB8.0 и VB 9.0	C#, VB.NET и Java	так (XMI 2.1)
Rational Rose	Rational Software Corporation	Windows, Sun SPARC stations (UNIX, Solaris, SunOS), Hewlett-Packard (HP UX), IBM RS/6000	ні	C++, Smalltalk, PowerBuilder, Ada, SQLWindows и ObjectPro	C++, Smalltalk, PowerBuilder, Ada, SQLWindows и ObjectPro	так
Visual Paradigm	Visual Paradigm	Linux, Mac OS X, Windows	ні	C#, VB .NET, Object Definition Language (ODL), Flash ActionScript, Delphi, Perl, Objective-C, Ruby	Java, C++, CORBA IDL, PHP, XML Schema, Ada, Python ,Java class, .NET dll u exe, JDBC	так (XMI 1.0, 1.2, 2.1)

До особливих переваг Visual Paradigm у порівнянні з іншими системами слід віднести також широкі можливості для створення моделей, так зокрема аналітикові в

межах одного проекту надається можливість опису вимог, з одночасним використанням стандартів і нотації моделювання: UML, BMPN, OMG, Mind Maps і ArchiMate.

Приклад створення діаграми прецедентів у системі Visual Paradigm

Як приклад наведемо процес створення діаграми прецедентів автоматизованої інформаційної системи управління ВНЗ модуль «Управління інформацією про співробітника».

Для заповнення діаграми потрібно розмістити на ній акторів та варіанти використання, користуючись блоками Actor Grid та Use Case Grid.

Для нашої предметної області вводимо наступних акторів:

Таблиця 2

Актор	Стислий опис
Фахівець по роботі зі співробітниками	Співробітник,який безпосередньо спілкуеться з співробітником та працює з даними про нього
Фахівець з організації нормативно-правової діяльності	Співробітник, який слідкуе за виходом та реалізацією наказів

Опишемо, які можливості повинна надавати система:

- актор «Фахівець по роботі зі співробітниками» використовус систему для створення, редагування даних про роботу співробітника та управління особистою інформацією про співробітника навчального закладу;
- актор «Фахівець з організації нормативно-правової діяльності» використовуе систему для внесення, видалення та перегляду існуючих наказів. Зміна наказів може відбуватися до проведения наказів в базі.

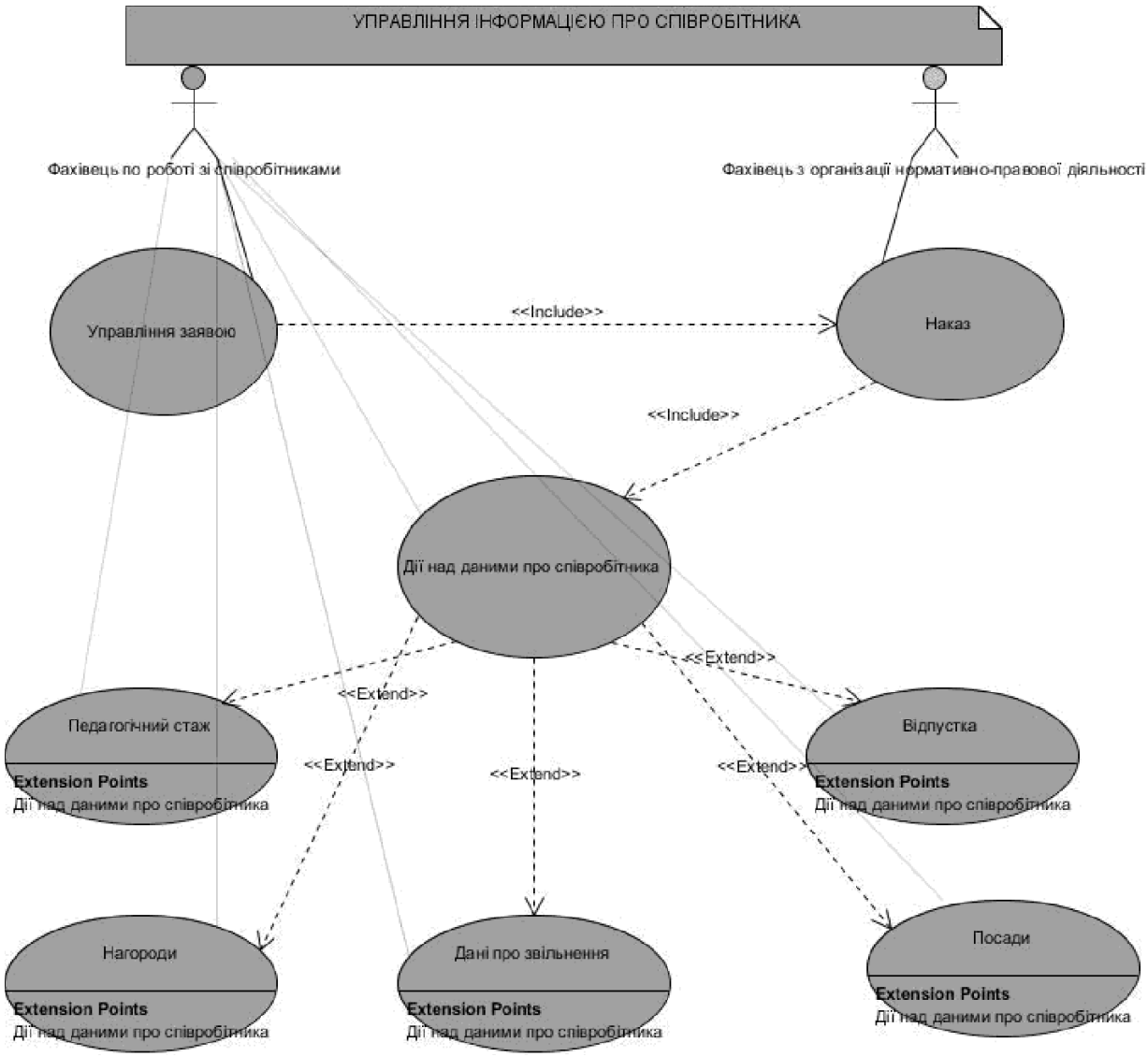


Рис.1. Діаграма прецедентів

На основі описаних можливостей визначаємо прецеденти:

Таблиця 3

Опис прецедентів та їх характеристик

Прецедент	**Стислий опис**
Дії над даними про співробітника	Запускається фахівцем по роботі зі співробітниками. Дозволяг вносити, змінювати та переглядати дані про співробітника на основі наказів
Заява	Дозволяс додавати, змінювати чи видаляти заяву про прийняття на роботу
Дані про звільнення	Інформація про причину звільнення
flосади	Список можливих посад, які займають співробітники

Наказ	Запускасться фахівцем з організаціі нормативно-правовоі діяльності. Дозволяе вносити, змінювати (поки вони не проведені), видаляти та переглядати існуючі накази. Тако›к потрібно передбачити довідник по наказах
Відпустка	Опис кількості фактичних вихідних днів, які надані співробітнику
Нагороди	Облік нагород, премій, відзнак і т.д.
fледагогічний стаж	Нотування стороннього педагогічного стажу

Розглянемо відношення між акторами та прецедентами. На мові UML можливий лише один тип відношення між актором та прецедентом — *відношення комунікаціі'.* Тому всіх акторів ми зв'язусмо з прецедентами відношенням *Association.* Оскільки інший тип відношень тут задати не можна, то стереотип *communicate* можна не вказувати.

Відношення між всіма прецедентами, крім Заява, Наказ та Дїі над даними про співробітника — *відношення розширення,* оскільки коли актор *Фахівець по роботі зі співробітниками* працює з даними про співробітника (оформляе, змінює і т.д.), то при цьому він не управлял інформаціею про накази, яка зв'язана із замовленням.

Відношення між прецедентами Заява, Наказ та Дії над даними про співробітника — відношення включения, оскільки для створення нового співробітника обов'язково потрібно створити заяву на прийняття на роботу і відповідний наказ.

Потік подій для прецедентів будемо описувати за наступним шаблоном.

- X.1 передумови;
- X.2 головний потік;
- X.3 підпотоки;
- X.4 альтернативні потоки;
- X.5 постумови,

де X — число від 1 до кількості прецедентів.

Такий опис потрібен для аналізу діаграми і майбутнього програмування системи.

Потік подій для прецеденту «Заява»

1.1. Передумови

Повинна виконуватись тільки після підкріплення до прецеденту «Наказ», при чому видалення заяви буде після цього неможливим.

1.2. Головний потік

Прецедент починас виконуватися, коли фахівець підключасться до системи і вводить свос ім'я і пароль. Система перевіряс правильність пароля і виводить можливі варіанти дій: додати, змінити, переглянути чи вийти. Якщо вибрана операція додати, S-1: виконуеться потік додати нову заяву.

Якщо вибрана операція змінити, S-2: виконусться потік змінити даних в заяві (про заяву).

Якщо вибрана операція переглянути, S-3: виконусться потік переглянути дані про заяву.

Якщо вибрана операція вийти, прецедент завершуеться.

1.3. Підпотоки

S-1: додати нову заяву.

Система відображае вікно діалогу, яке містить *пот* вводу даних для нової заяви. Фахівець заповнюс поля: кому належить заява, дата написания заяви, на чие ім'я, призначення, на який термін, кількість ставки і т.і. Система запам'ятовус введені дані. Далі заява виводиться на друк. Потім прецедент розпочинасться спочатку.

S-2: змінити дані заяви

Система відображае вікно діалогу, яке містить список заяв і поле для вводу номера (ідентифікатора) заяви. Фахівець вибирае необхідну заяву зі списку чи вводить іі номер (ідентифікатор) у відповідне поле. Система відображас інформацію про дану заяву. Фахівець вносить потрібні зміни. Система запам'ятовуе введені дані. Потім прецедент розпочинаеться спочатку.

S-3: переглянути дані про заяву

Система відображае вікно діалогу, що містить список заяв і поле для вводу номера (ідентифікатора) заяви. Менеджер вибирас необхідну заяву зі списку чи вводить іі номер (ідентифікатор) у відповідне поле. Система відображае інформацію про заяву (саму заяву). Коли фахівець перегляне інформацію, прецедент

Видалення заяви можливе до того моменту поки вона не підкріплена до наказу.

1.4. Альтернативні потоки

1: введено неправильне ім'я чи пароль. Користувач повинен повторити введення чи завершити прецедент.

2: заповнені не всі поля. Фахівець повинен заповнити пусті поля чи завершити

прецедент.

3: введено неправильний номер (ідентифікатор) заяви. Фахівець повинен повторити введення чи завершити прецедент.

За аналогічним сценарієм мають бути описані потоки подій для інших прецедентів.

Після опису моделі прецедентів переходимо до побудови та опису наступних моделей.

Висновки. Обсяги і складність сучасних інформаційних систем призвели до того, що UML- проектування виявляється обов'язковою складовою процесу розробки й передує етапу реалізації. Отримані в такий спосіб діаграми е основою проектованого програмного забезпечення й подаються у вигляді формальних інформаційних моделей. Розглянутий фрагмент прикладу показує лише деякі прийоми проектування програмного забезпечення за допомогою методів UML.

Огляд методологій проектування складних систем

Програма дисципліни передбачає вивчення *методології структурного аналізу* і *об'єктної методології* проектування інформаційних систем (ІС).

У Стандарті ISO / IEC 2382–1 наведено таке визначення: «*Інформаційна система* – система обробки інформації, що працює спільно з організаційними ресурсами, такими як люди, технічні засоби та фінансові ресурси, які забезпечують і розподіляють інформацію».

Структурна та об’єктна методології проектування ІС відповідають принципам системного аналізу та закладають основу блочно–ієрархічного підходу до проектування. Кожен рівень ієрархії визначається в результаті декомпозиції. У той же час декомпозиція проводиться за різними правилами. У методології

Методологію структурного аналізу доцільно застосовувати для проектування різних процесів (технологічних та бізнес- процесів, інструкцій та ін.), Тобто де розглядаються дії (роботи).

Об'єктна методологія використовується при проектуванні програмного забезпечення, моделюванні пристроїв у вигляді сукупності об'єктів, в процесі взаємодії яких через передачу повідомлень відбувається виконання необхідних функцій.

Методологія структурного аналізу

Structured Analysis and Design Technique (SADT) (Технологія структурного аналізу і

проектування) – одна з найвідоміших методологій аналізу та проектування систем, яка була розроблена в 70–і роки 20 століття. В даний час ця методологія трансформувалась в методологію *IDEF0 (Integrated DEFinition).*

Опис системи за допомогою IDEFO називається *функціональною моделлю*. Основним елементом у моделі є *діаграма*. Модель може об'єднувати кілька діаграм в одну ієрархію. Чим глибше діаграма знаходиться в ієрархії, тим більше вона деталізована, тобто тим більш докладно відображає дані або активності системи, або блоку.

Процес моделювання включає збір інформації про досліджувану область, документування отриманої інформації і представлення її у вигляді моделі і уточнення моделі за допомогою ітеративного рецензування. Крім того, цей процес підказує цілком певний шлях виконання узгодженої та достовірної структурної декомпозиції, що є ключовим моментом у кваліфікованому аналізі системи. Методологія IDEFO унікальна в своїй здатності забезпечити як графічну мову, так і процес створення несуперечливої і корисної системи описів.

IDEFO є методологією в повному розумінні, тому що вона об'єднує ітеративний процес створення моделі, нотації, мову посилань для діаграм, мову функцій моделей з графічною мовою опису системи, а також рекомендації щодо реалізації аналітичних проектів. Нотації, які керують конфігурацією, гарантують, що нові діаграми будуть коректно вбудовані в ієрархічну структуру моделі. Мова посилань в IDEFO, правила скорочень для посилань, адресованих до окремих частин діаграми, полегшують оформлення зауважень при рецензуванні моделі. Мова функцій дозволяє декларативно визначати правила роботи системи, що часто

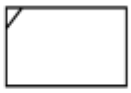
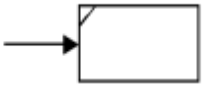
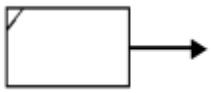
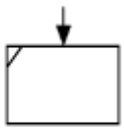
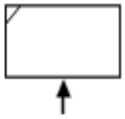
7

Н.Л. Дорош, Ю.В. Бабенко «Проектування організаційних і технологічних інформаційних управляючих систем»

є особливо важливим завершальним кроком в описі системи. Методологія IDEFO реалізована в програмах *BPWIN, RAMUS.*

У таблиці 2.2 наведені основні «будівельні блоки» для діаграм IDEF0.

Таблиця 2.2 - Основні «будівельні блоки» для діаграм IDEF0

№	Назва	Опис елементу IDEF0 діаграми	Графічне представлення
1	2	3	4
1	Модуль поведінки.	Об'єкт служить для опису функцій (процедур, робіт), які виконуються підрозділами / співробітниками підприємства	
2	Стрілка входу (input)	Стрілка описує вхідні документи, інформацію, матеріальні ресурси, необхідні для виконання функції.	
3	Стрілка виходу (output)	Стрілка описує вихідні документи, інформацію, матеріальні ресурси, що є результатом виконання функції.	
4	Стрілка управління (control)	Стрілка описує керуючий вплив, наприклад розпорядження, нормативний документ і т.д. У нотації IDEF0 кожна процедура повинна обов'язково мати не менше однієї стрілки зверху, що відбиває керуючий вплив.	
5	Стрілка механізму (mechanism)	Стрілка знизу описує т.зв. механізми, тобто ресурси, необхідні для виконання процедури, але не змінюють в процесі її виконання свій стан. Приклади: співробітник, верстат.	

Варто відзначити, що у моделі всі роботи і стрілки повинні бути іменовані. Кількісний аналіз діаграм Для проведення кількісного аналізу діаграми перерахуємо показники моделі:

• кількість блоків на діаграмі – N;

• рівень декомпозиції діаграми – L;

• збалансованість діаграми – B;

• число стрілок, що з'єднуються з блоком – A.

Даний набір факторів відноситься до кожної діаграми моделі. Далі будуть перераховані рекомендації з бажаним значенням факторів діаграми.

Необхідно прагнути до того, щоб кількість блоків на діаграмах нижніх рівнів була б нижче кількості блоків на батьківських діаграмах, тобто зі збільшенням рівня декомпозиції зменшувався б коефіцієнт N \ L. Таким чином, спадання цього коефіцієнта говорить про те, що в міру декомпозиції моделі функції повинні спрощуватися, отже, кількістьблоків повинна зменшуватися.

Діаграми повинні бути збалансовані. Це означає, що в рамках однієї діаграми не повинно відбуватися ситуації, коли в роботу вхідних стрілок і стрілок управління значно більше, ніж які виходять. Слід зазначити, що дана рекомендація може не виконуватися в моделях, що описують виробничі процеси. Наприклад, при описі процедури складання в блок може входити безліч стрілок, що описують компоненти виробу, а виходити одна стрілка – готовий виріб.

Введемо коефіцієнтзбалансованоїдіаграми:

$$K_b = \left| \frac{\sum_{I=1}^{N} A_I}{N} - \max_{I=1}^{N}(A_I) \right|$$

Необхідно прагнути, щоб Kb був мінімальний для діаграми. Крім аналізу графічних елементів діаграми необхідно розглядати найменування блоків. Для оцінки імен складається словник елементарних (тривіальних) функцій модельованої системи. Фактично в даний словник повинні потрапити функції нижнього рівня декомпозиції

діаграм. Наприклад, для моделі БД елементарними можуть бути функції "знайти запис", "додати запис в БД", в той час як функція "реєстрація користувача" вимагає подальшого опису.

Після формування словника і складання пакету діаграм системи необхідно розглянути нижній рівень моделі. Якщо на ньому виявляться збігу назвах блоків діаграм і слів зі словника, то це говорить, що достатній рівень декомпозиції досягнутий. Коефіцієнт кількісно відображає даний критерій, можна записати як L * C –добуток рівня моделі на число збігів імен блоків з словами зі словника. Чим нижче рівень моделі (більше L),тимцінніше збіги.

Діаграмипотоків даних (DataFlowDiagrams)

Діаграми DFD можна використовувати як доповнення до діаграм IDEF0 для опису документообігу та обробки інформації. Ці діаграми представляють мережу пов'язаних між собою робіт.

DFD описує:

- •функціїобробки інформації (роботи);
    - документи (стрілки, arrow), об'єкти, співробітників або відділи, які беруть участь вобробціінформації;
    - зовнішні посилання (external reference), які забезпечують інтерфейс з зовнішніми об'єктами, щознаходяться за межами модельованоїсистеми;
    - таблиці для зберігання документів (сховища даних,data store).

Потоки даних є механізмами, що використовуються для моделювання передачі інформації (або фізичних компонентів) з однієї частини системи в іншу. Потоки зображуються на діаграмі іменованими стрілками, орієнтація яких вказує напрямок руху інформації. Стрілки можутьпідходити добудь–якої грані прямокутникароботи і можуть бути двонаправленими для описувзаємодії типу"команда–відповідь".

Призначення *процесу* полягає в продукуванні вихідних потоків із вхідних відповідно до дії, що задається ім'ям процесу. Кожен процес повинен мати унікальний номер для посилань на нього всередині діаграми. Цей номер може використовуватися спільно з номеромдіаграми для отримання унікальногоіндексупроцесуувсій моделі.

Сховище даних дозволяє на певних ділянках визначати дані, які будуть зберігатися в пам'яті між процесами. Фактично сховище представляє "зрізи" потоків даних у часі.

Інформація, яку воно містить, може використовуватися в будь–який час після її визначення, при цьому дані можуть вибиратися в будь–якому порядку. Ім'я сховища має ідентифікувати його вміст. У випадку, коли потік даних входить в сховище або виходить з нього і його структура відповідає структурі сховища, він повинен мати те ж саме ім'я.

Зовнішня сутність – це сутність поза контекстом системи, що є джерелом або приймачем даних системи. Передбачається, що об'єкти, що представлені такими вузлами, не повинні брати участь ні в якій обробці. Зовнішні сутності зображуються у вигляді прямокутника з тінню і зазвичай розташовуються по краях діаграми. Одна зовнішня сутність може бути використана багаторазово на одній, або декількох діаграмах.

БІБЛІОГРАФІЯ

2 Винчугова А.А. Методы и средства концептуального проектирования информационных систем: сравнительный анализ стуктурного и объектно-ориентированого подходов / А.А.Винчугова // Прикладная информатика №1(49). - 2014. - С.56-66

3 MS Visio [Електронний ресурс] - Режим доступу: http://office.microsoft.com/en-us/FX010857981033.aspx

4 Dia [Електронний ресурс] - Режим доступу: http://live.gnome.orgЮia

5 Enterprise Architect [Електронний ресурс] - Режим доступу: http://www.sparxsystems.com/products/ea/index.html

6 Umbrello [Електронний ресурс] - Режим доступу: http://uml.sourceforge.net/

9.ArgoUML [Електронний ресурс] - Режим доступу: http://argouml.tigris.org/

7 Visual Paradigm [Електронний ресурс] - Режим доступу: http://visua1-paradigm.com/