Висновок

В контексті реляційних баз даних ми виділили два основні незалежні часові виміри: дійсний час та час транзакції. Збереження даних в розрізі цих вимірів є доцільне в багатьох предметних областях, навіть у музейній справі, яка не відзначається особливою динамікою. Однак в багатьох випадках дані можуть пов'язуватися з іншими часовими вимірами, семантика яких зрозуміла лише користувачеві бази даних. В таких випадках предметні області вимагають ретельного дослідження на предмет семантики часових параметрів.

1. Жежнич П.І. Реляційні бази даних з часовою інтерпретацією. //Вісн. ДУ "Львівська політехніка", Інформаційні системи та мережі – №383. – 1999. 2. Жежнич П.І., Пелещишин А.М. Історична інформаційна система "Електронний музей" // Вісн. НУ "Львівська політехніка", Інформаційні системи та мережі. – №406. – 2000. 3. Jensen C.S., Clifford C., Elmasri R., Gadia S.K., Hayes P., Jajodia S. A consensus glossary of temporal database concepts //Technical Report R 93-2035, Dept. of Mathematics and Computer Science, Inst. for Electronic Systems, Denmark, Nov. 1993. 4. Snodgrass R.T., Ahn I. Temporal databases. //Computer. – 1986. Vol.19, №.9, P. 35 – 42, Sept.

УДК 681.3

А. В. Катренко
НУ "Львівська політехніка",
кафедра "Інформаційні системи та мережі"

ЗАСТОСУВАННЯ ТЕХНОЛОГІЙ ТА ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

In this article: the peculiarities and the requirements to the methodologies and technologies of information systems design are considered; the problems that arise at the early design stages and during the business process reengineering using CASE-tools are presented. The appropriate methods to solve these problems are proposed as well.

В статті розглянуті вимоги до методологій та технологій проектування інформаційних систем; здійснений аналіз структурного та об'єктного підходу та особливостей застосування CASE-засобів до процесів бізнес-реінженерії. Запропоновані методи для розв'язування проблем проектування та реінженерії на різних етапах циклу життя інформаційних систем.

Методології проектування інформаційних систем (ІС) та їх реалізації у вигляді технологій і інструментальних засобів незалежно від підходів, що використовуються, скеровані на виконання наступних основних вимог [1-3]:

- декомпозиція процесу проектування на основні стадії – аналіз, проектування, реалізація (кодування, тестування, впровадження), супровід;
- забезпечення колективної роботи над проектом (наявність засобів координації та управління колективом розробників);
- скорочення циклу розробки ІС, щоб він був значно коротшим, аніж цикл життя системи;
- підвищення гнучкості інструментальних засобів для врахування змін вимог до ІС.

Особливої уваги потребують початкові стадії процесу – аналіз та проектування, тому що виправлення помилок, зроблених на цих стадіях, під час реалізації наступних вартуватиме на порядок більше, ніж у випадку їх своєчасного виправлення на початкових стадіях. Для забезпечення цього вкрай важливою є наявність ефективних засобів автоматизації на початкових етапах реалізації проекту ІС. Сучасні корпоративні ІС не можуть бути реалізованими однією особою, а тому необхідно мати засоби, що забезпечуватимуть ефективну колективну роботу.

Характерним для сучасних компаній є високий динамізм, який виявляється в постійній перебудові власних бізнес-процесів, переході на нові технології праці, що викликає необхідність максимального скорочення тривалості циклу розробки ІС (інакше може статися, що спроектована система призначалася для використання в таких умовах, яких вже не існує). З цієї ж причини – змін зовнішніх умов та вимог – інструментальні засоби повинні забезпечувати можливість внесення змін не лише на ранніх, але й на пізніх стадіях проектування. Необхідність реінженерії пов'язана з високою динамічністю сучасного ділового світу. Неперервні і досить істотні зміни в технологіях, ринках збуту і потребах клієнтів стали звичайним явищем, і компанії, прагнучи зберегти свою конкурентоспроможність, змушені постійно перебудовувати корпоративну стратегію і тактику.

Методології, технології й інструментальні засоби проектування складають основу проекту будь-якої ІС. Методологія реалізується через конкретні технології і підтримуючі їх стандарти, методики й інструментальні засоби, що забезпечують виконання процесів ЦЖ.

Технологія проектування визначається як сукупність трьох складових:

- покрокової процедури, що визначає послідовність технологічних операцій проектування;
- критеріїв і правил, використовуваних для оцінювання результатів виконання технологічних операцій;
- нотацій (графічних і текстових засобів), що використовуються для описання системи, що проектується.

Технологічні інструкції, що становлять основний зміст технології, повинні складатися з описання послідовності технологічних операцій, умов, залежно від яких виконується та або інша операція, і описань самих операцій (рис. 1).

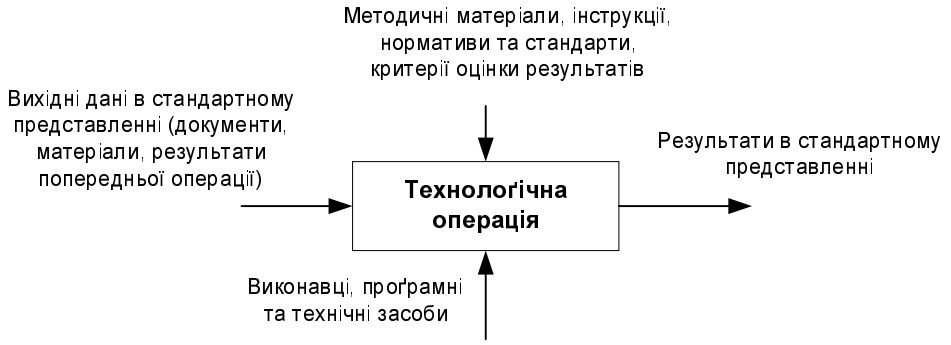

Рис. 1. *Представлення технологічної операції проектування*

Технологія проектування, розроблення і супроводу ІС повинна відповідати таким загальним вимогам [1,4]:

- технологія повинна підтримувати повний ЦЖ ІС;
- технологія повинна забезпечувати гарантоване досягнення цілей розроблення ІС із заданою якістю й у встановлений час;

- технологія повинна забезпечувати можливість виконання значних проектів у вигляді підсистем;
- технологія повинна забезпечувати можливість ведення робіт із проектування окремих складових невеличкими групами (3-7 чоловік). Це обумовлено принципами керованості колективу і підвищення продуктивності за рахунок мінімізації числа зовнішніх зв'язків; технологія повинна забезпечувати мінімальний час одержання працездатної ІС;
- технологія повинна передбачати можливість керування конфігурацією проекту, ведення версій проекту і його складових, можливість автоматичного випуску проектної документації і синхронізацію її версій із версіями проекту;
- технологія повинна забезпечувати незалежність виконуваних проектних рішень від засобів реалізації ІС (систем керування базами даних, операційних систем, мов і систем програмування);
- технологія повинна підтримуватись комплексом узгоджених інструментальних засобів, що забезпечують автоматизацію процесів, які виконуються на всіх стадіях ЦЖ.

Перераховані чинники сприяли появі програмно-технологічних засобів, що реалізують CASE-технологію створення і супроводу ІС (CASE-засобів). CASE є результатом еволюційного розвитку з метою усунення недоліків та обмежень, що виникали при використанні структурних методологій проектування ІС 1960-1970-х років (складність розуміння, велика трудомісткість та вартість використання, складності внесення змін до проектних специфікацій та ін.) шляхом автоматизації та інтеграції засобів підтримання. Термін CASE (початково Computer Aided Software Engineering — а тепер Computer Aided System Engineering) використовується у дуже широкому контексті. Початкове значення терміну CASE, обмежене питаннями автоматизації розроблення лише програмного забезпечення ІС, тепер набуло нового сенсу, що охоплює процес розроблення складних ІС загалом. Сучасні інструментальні засоби, побудовані на ґрунті CASE-технології, підтримують процеси створення і супроводу ІС, зокрема аналіз і формулювання вимог, проектування прикладного ПЗ (застосувань) і баз даних, ґенерацію коду, тестування, документування, забезпечення якості, конфігураційне керування і керування проектом, а також інші процеси, тобто разом із системним ІС і технічними засобами утворюють повне середовище розроблення ІС.

Таким чином, CASE – це технологія системних аналітиків, розробників та програмістів, що дозволяє автоматизувати процес проектування та розроблення складних ІС. Інструментарій CASE використовується як для аналізу або перепроектування існуючих систем, так і для проектування нових. Внаслідок цього акценти в процесі проектування змістилися в бік покращення якості власне системно-аналітичних та проектних робіт початкових етапів, що привело до значного покращання проектів ІС загалом та суттєвого полегшення процесів супроводу та внесення змін в ці проекти.

Сьогодні CASE-технологія є основною технологією проектування та реінженерії ІС. Практично ні один серйозний проект ІС не виконується без застосування засобів CASE. Головна мета CASE полягає в тому, щоб відділити проектування ІС від кодування відповідного програмного забезпечення (ПЗ) та наступних етапів розробки, "заховати" від розробників деталі середовища розробки та функціонування ПЗ.Сучасна CASE-технологія дозволяє у наочній формі моделювати предметну область, аналізувати цю модель на всіх етапах розроблення і супроводу ІС і розробляти застосування відповідно до інформаційних потреб користувачів. При використанні CASE-технологій змінюються всі етапи циклу життя ІС, а найбільше — етапи аналізу та проектування. В сучасних CASE-системах використовуються методології як об'єктного, так і структурного аналізу та проектування, що ґрунтуються на наочних техніках побудови діаґрам з використанням для опису моделі ІС,

що проектується, ґрафів, діаґрам, таблиць та схем. Ці методології забезпечують строге та наочне відображення ІС, що починається з загальних аспектів і потім деталізується, набуваючи рис багаторівневої ієрархічної структури.

Перевагами CASE-технологій є:

- покращання якості ПЗ, що створюється за їх допомогою, за рахунок засобів автоматичного контролю, насамперед контролю проекту;
- зменшення часу, необхідного для створення прототипу майбутньої системи, що дозволяє зробити оцінку очікуваного результату на ранніх етапах;
- прискорення процесу аналізу та проектування;
- звільнення розробника від рутинної роботи, зосереджуючи його увагу на творчих аспектах проекту;
- підтримання розвитку та супровід розробки;
- підтримання технології повторного використання компонент проекту.

На даний час найбільшого поширення набули об'єктно-орієнтований та структурний підхід, а тому проаналізуємо можливості та доцільність застосування їх загалом та на окремих етапах проектування.

Структурна методологія проектування ІС

Суть структурного підходу до розробки ІС полягає в декомпозиції системи на функції, що автоматизуються: система розбивається на функціональні підсистеми, які у свою чергу діляться на підфункції, що підрозділяються на задачі і так далі. Процес розбиття продовжується аж до конкретних процедур. При цьому система, що автоматизується, зберігає цілісне представлення, у якому всі складові компоненти взаємно погоджені. При розробці системи "нагору" від окремих задач до системи загалом цілісність губиться, виникають проблеми при інформаційному узгодженні окремих компонентів [1].

Структурний аналіз і проектування – назва класу декількох формальних методів, що використовуються для аналізу і проектування систем. Хоча можливо розробити моделі діяльності поза структурним аналізом проекту, проте важливо зрозуміти контекст, у якому моделі діяльності зазвичай формуються. На додаток до моделей діяльності, структурний аналіз включає інформаційні моделі, що моделюють дані, і діаґрами переходів (state-transition diagrams – STD), що моделюють поведінку системи в часі, а також розширення реального часу.

Основний принцип структурного аналізу і проектування – повний аналіз поточної системи – є необхідним для проектування оптимального рішення. Значна кількість даних збирається аналітиком, що бере інтерв'ю у фахівців (експертів) із певної області знань – осіб, які добре інформовані щодо системи, що нас цікавить. Хоча існують методи моделювання діяльності (наприклад, IDEF3), щоб полегшити збирання інформації від фахівців з проблемної області, проте центральна роль аналітика, особливо на стадіях аналізу та попереднього проектування, залишається значною.

Спочатку моделі розробляються для документування поточних характеристик системи (відомі, як AS-IS — "як є" моделі). Ці моделі пізніше використовуються як передумова для розробки моделей проектування (відомі як TO-BE — "як повинно бути" моделі). Вони також використовуються як еталонний тест, з якими TO-BE моделі співставляються, щоб гарантувати, що запропонований проект – дійсно покращений [3].

На додаток до управління проектом запропонованої системи, TO-BE моделі використовуються для планування завантаження, складання бюджету і отримання ресурсів. Моделі також використовуються, щоб одержати проектний план реалізації, що визначає, як система перетворена від AS-IS ситуації в TO-BE ситуацію.

Потрібно порівнювати прибуток від розробки AS-IS моделей з вартістю та часом розробки. Ділова література переповнена прикладами існуючих систем, що розв'язують хибну проблему, яку може допомогти зрозуміти AS-IS моделювання. З іншого боку, якщо

проблема в загальному добре зрозуміла, то рентабельність AS-IS моделювання є менш очевидною.

Структурний аналіз – це метод дослідження системи, який починається із загального розгляду системи з наступною деталізацією, що має ієрархічну багаторівневу структуру. Для методів цього типу характерним є:

- розбиття на рівні абстракцій з обмеженням числа елементів кожного рівня (від 3 до 6-7);
- обмежений контекст, що включає лише суттєві на кожному рівні деталі;
- двоїстість даних та операцій над ними;
- використання строгих формальних правил запису;
- послідовне наближення до остаточного результату.
- В основі методології структурного підходу лежать наступні принципи [1,5]:
- Принцип декомпозиції – "поділяй і пануй" – розв'язання складних завдань досягається шляхом розбиття їх на множину менших незалежних завдань, які можуть бути зрозумілими та розв'язаними.

Принцип ієрархічного впорядкування – будова частин, менших незалежних завдань є суттєвою для розуміння системи. Розуміння проблеми різко підвищується за умови орґанізації її складових в деревовидні ієрархічні структури.

Принцип абстраґування – в першу чергу виділяються суттєві аспекти системи та нехтується несуттєвими з метою представлення проблеми в простому загальному вигляді.

Принцип формалізації – для розв'язування проблеми необхідний строгий методичний підхід.

Принцип маскування – на кожному конкретному етапі маскується несуттєва власне на цьому етапі інформація: кожна складова "знає" лише необхідну їй інформацію.

Принцип концептуальної спільності полягає у використанні єдиної філософії на всіх етапах життєвого циклу ІС (структурний аналіз – структурне проектування – програмування).

Принцип мінімальності – здійснюється контроль на наявність зайвих елементів.

Принцип несуперечливості полягає в обґрунтованості та узгодженості елементів.

Принцип логічної незалежності полягає в особливій увазі до логічного проектування з метою забезпечення незалежності від фізичного проектування.

Принцип незалежності даних – моделі даних повинні бути проаналізовані та спроектовані незалежно від процесів їх логічного опрацювання, а також від фізичної структури та розподілу.

Принцип структурованості даних – дані повинні бути структуровані та ієрархічно орґанізовані.

Принцип доступу остаточного користувача – користувач повинен мати засоби безпосереднього доступу до БД, які він міг би використати безпосередньо – без програмування.

Об'єктно-орієнтована методологія проектування ІС

Об'єктний підхід (ОП) принципово відрізняється від тих підходів, що пов'язані з традиційними засобами структурного аналізу, проектування і програмування. Це не означає, що об'єктний підхід вимагає відмови від всіх раніше знайдених і випробуваних методів і прийомів, навпаки, нові елементи завжди ґрунтуються на попередньому досвіді. Концепції та ідеї, на яких ґрунтується ОП, постали набагато раніше, аніж з'явився сам термін, і беруть свій початок в галузі моделювання систем (поняття аґреґату є близьким до поняття об'єкта, ідеї мови моделювання SIMULA були покладені в основу суто об'єктної мови SMALLTALK та використовувались в мові ADA).

ОП створює істотні зручності, які не можуть бути забезпечені за інших умов. Найважливішим є те, що ОП дозволяє створювати системи, які ґрунтуються на основних принципах добре структурованих складних систем.

Таким чином, ОП є не чим іншим, як певним способом структуризації уявлень про систему, що має ряд суттєвих переваг [2,6]:

- дозволяє повноцінно використати можливості об'єктних і об'єктно-орієнтованих мов програмування;
- використання ОП істотно підвищує якість розробки як системи загалом, так і її фраґментів. Об'єктно-орієнтовані системи виявляються компактнішими, ніж необ'єктно-орієнтовані еквіваленти. А це означає не лише зменшення обсягу коду програм, але і здешевлення проекту і більшу зручність в плануванні розробок;
- використання ОП веде до побудови систем на основі стабільних проміжних описів, що спрощує процес внесення змін. Це дає системі можливість розвиватися поступово і не приводить до повної її переробки у випадку істотних змін вхідних вимог;
- зменшує ризик в розробці складних систем передусім тому, що процес інтеґрації розтягується в часі життєвого циклу системи.

Головним в понятті об'єкта є об'єднання ідей абстраґування даних та алгоритмів. Внаслідок цього акцент переноситься на конкретні характеристики фізичної чи абстрактної системи, що є предметом створення програмного забезпечення та моделювання (хоча в багатьох випадках поряд з об'єктами, які відповідають елементам реальної системи, розробляються об'єкти та класи, які не відповідають сутностям реальної системи). Об'єктний підхід реалізується в **об'єктно-орієнтованому аналізі** (OOA – Object Oriented Analyses), **об'єктно-орієнтованому проектуванні** (OOD – Object Oriented Design) та **об'єктно-орієнтованому програмуванні** (OOP – Object Oriented Programming).

В ідеалі за результатами OOA повинні формуватися моделі, на яких ґрунтується OOD. OOD утворює основу для остаточної реалізації системи з використанням методології OOP. Значною перевагою об'єктного підходу є те, що він може реалізуватися й на вищих рівнях абстракції, тобто на кожному рівні можна виділити ґрупи об'єктів, що тісно взаємодіють для розв'язування задачі вищого рівня абстраґування.

Об'єктно-орієнтований аналіз має на меті створення моделей, близьких до дійсності, з використанням об'єктно-орієнтованого підходу. Іншими словами, це методологія, в якій вимоги формуються на основі понять класів та об'єктів, що складають зміст словника предметної області. При цьому способи формування такого словника, як правило, знаходяться за межами ОП [4].

Об'єктно-орієнтоване проектування – це методологія проектування, що поєднує в собі об'єктну декомпозицію та прийоми представлення як логічної і фізичної, так і статичної та динамічної моделей системи.

Логічна структура відображається у вигляді абстракцій типу класів та об'єктів, і саме *підтримка об'єктно-орієнтованої декомпозиції є вирізняльною особливістю об'єктно-орієнтованого проектування порівняно з структурним проектуванням*. Таким чином, ОП виник в результаті еволюційного, а не революційного процесу, тобто він успадковує багато особливостей попередніх методологій.

Об'єктно-орієнтоване проектування ґрунтується на таких основних принципах: **абстраґування; обмеження доступу; модульність; ієрархічність; типізація; паралелізм; стійкість.**

Об'єднані в OOD, ці відомі принципи дозволили розв'язувати складні задачі проектування інформаційних систем. OOD розвинувся природним чином внаслідок зміщення ваги від програмування окремих деталей до програмування систем, а також розвитку та вдосконалення мов програмування високого рівня.

Об'єктно-орієнтоване програмування базується на представленні програми у вигляді множини об'єктів, кожен з яких належить до певного класу, а класи утворюють ієрархію, побудовану на принципі успадковування. Таким чином ООР використовує в як базові конструкції об'єкти, а не алгоритми, об'єкти є реалізаціями певних класів, класи будуються на принципі успадковування, і система функціонує шляхом опрацювання та ґенерації подій, що викликають ту чи іншу реакцію певних об'єктів.

Межі між стадіями об'єктно-орієнтованого аналізу і об'єктно-орієнтованого проектування розмиті, але задачі, які ними вирішуються, визначаються достатньо чітко [2]. В процесі об'єктно-орієнтованого аналізу ми моделюємо проблему, визначаючи класи і об'єкти. При об'єктно-орієнтованому проектуванні ми знаходимо абстракції і механізми, що забезпечують правильну поведінку нашої моделі. Таким чином, при розробці системи процес проектування продовжує роботу, розпочату на стадії аналізу.

Модель ООА описує об'єкти, що утворюють разом із різноманітними структурними та комунікаційними відношеннями окрему сферу застосування. Модель ООА виконує дві функції.

По-перше, вона використовується для формалізації представлення реального світу, у який буде вбудована модель ІС: визначає класи та об'єкти, що виступають як основні структурні елементи орґанізації системи, і різноманітні правила та/або обмеження, котрі накладаються реальним світом на ІС.

По-друге, вона встановлює, яким чином об'єкти взаємодіють між собою у процесі функціонування ІС.

Згідно з [4] процес проектування розглядається як макропроектування (діяльність всього колективу в масштабі від тижнів до місяців) та мікропроектування (щоденна праця окремого розробника чи невеликого колективу).

В макропроектуванні зберігаються традиційні фази аналізу та проектування з концентрацією на управлінні ризиком та виявленні загальної архітектури системи як двох компонентах, що мають вирішальне значення з точки зору строків, повноти та якості проекту. Макропроектування складається з таких дій:

- Виявлення суті вимог до проґрамного продукту (концептуалізація);
- Розроблення моделі поведінки системи, яка вимагається (аналіз);
- Створення архітектури для реалізації (проектування);
- Ітеративне виконання проекту (еволюція);
- Керування еволюцією продукту в процесі експлуатації (супровід).

Мікропроцес натомість характеризується тим, що традиційні фази аналізу та проектування є взаємно пов'язаними, переплетеними між собою, і складається з таких видів діяльності:

- Виявлення класів та об'єктів певного заданого рівня абстрагування;
- Виявлення семантики цих класів та об'єктів;
- Виявлення зв'язків між цими класами та об'єктами;
- Специфікація інтерфейсу та реалізація цих класів та об'єктів.

Сучасною реалізацією ОП є методологія RUP (Rational Unified Process) фірми Rational Rose, в якій декларується інструментальна підтримка всіх етапів життєвого циклу розробки ІС та ПЗ [2]. Структура методології RUP наведена на рис. 2.

Розрізняється динамічна та статична структура процесу проектування [6]. Динамічна структура поділяється на фази: початкову; розробку; реалізацію; розвиток. Кожна з фаз може складатися з декількох етапів.

До складу статичної структури входять базові та допоміжні процеси. Множину базових процесів утворюють такі: бізнес-моделювання; управління вимогами; аналіз і проектування; реалізація; тестування; розгортання. Допоміжними процесами є: управління конфігураціями; управління проектом; інструментальне середовище.

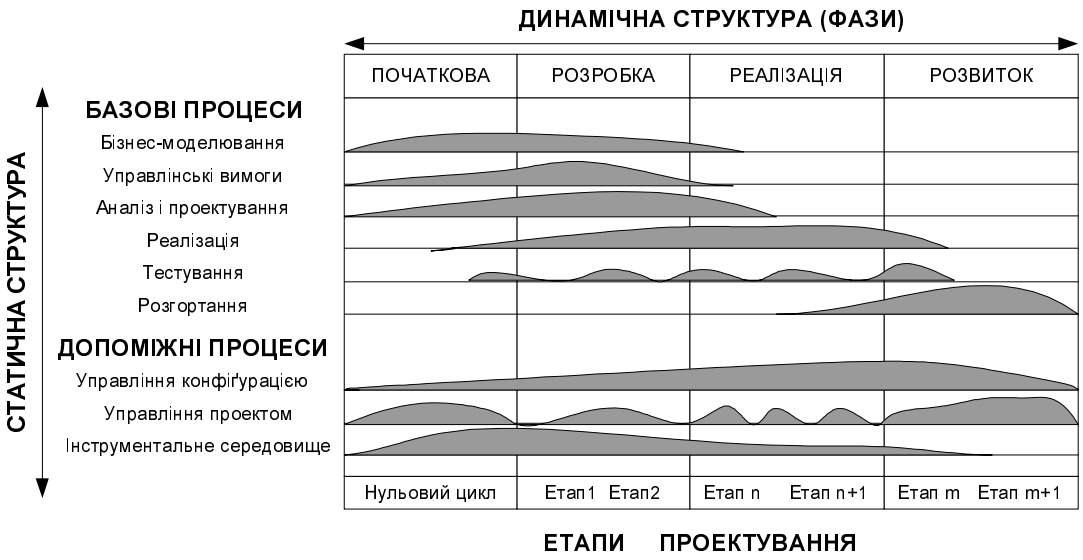

Рис. 2. *Структура методології Rational Unified Process (RUP)*

В RUP добре реалізований ітераційний підхід до створення проектів. Усі розробники, що входять до колективу, повинні бути інформовані про рішення, прийняті протягом усього процесу розробки. Щоб забезпечити взаємодію співвиконавців, у RUP використовуються поняття «вимога» і засоби роботи з ними. Це ключові поняття, навколо них зібрані всі елементи технологічного оснащення, до складу яких входять наступні.

RequisitePro — інструмент для реєстрації, класифікації і керування вимогами. Можливість відстеження залежностей вимог одна від одної дозволяє візуально визначати схожі вимоги в межах проекту. Зв'язки між вимогами дозволяють визначити ті вимоги, які потрібно додатково переглянути з появою змін. Тим самим упорядковується процес внесення й опрацювання змін. Атрибути вимог допомагають установлювати пріоритети, сортувати і обирати вимоги, тобто вся інформація про вимоги консолідована воєдино і відразу видно, хто, як і чому змінив вимогу.

Rational Rose — інструмент візуального проектування. Для створення моделей використовується стандартна нотація — уніфікована мова моделювання UML. Робота в Rose ґрунтується на використанні репозиторію і побудові діаґрам. Rational Rose містить засоби автоматичної ґенерації вихідних кодів програм. Одним з його переваг є наявність ефективних засобів зворотного проектування, що дозволяють автоматично будувати моделі по вихідних текстах програм, що дає змогу виконувати перепроектування успадкованих систем, включати в моделі бібліотечні класи і компоненти, використовувати в проекті раніше створені модулі.

Rational ClearCase — засіб конфігураційного керування при створенні великих ІС масштабу підприємства. Rational ClearCase використовується для підтримки розробки великих інформаційних систем, відслідковує зміни аж до кожного файлу і каталогу, полегшуючи керування повними історіями анотованих версій вихідних і двійкових кодів, створенням документації, бібліотек і об'єктів, проведенням тестування й інших параметрів ІС.

Rational ClearQuest — інструмент для керування запитом на зміни. Дає можливість відслідковувати і керувати всією діяльністю, пов'язаною зі змінами, що відбуваються протягом усього циклу життя, включаючи процес виправлення помилок, виявлених

тестувальниками. ClearQuest дозволяє добудовувати процеси внесення змін у ІС відповідно до потреб конкретної організації.

Rational TeamTest — засіб автоматизованого тестування застосувань. Його використання дозволяє охопити всі стадії тестування, включаючи планування, проектування тестів, проведення системного тестування, у тому числі функціонального і регресійного тестування.

PerformanceStudio — засіб навантажувального тестування застосувань; дозволяє оцінити, як система буде працювати в реальних умовах при різних конфігураціях апаратно-програмної платформи, виміряти час реакції системи на запити остаточного користувача, виявити "вузькі місця" у системі.

Нотації структурного підходу

Для цілей моделювання інформаційних систем взагалі і структурного аналізу зокрема, використовуються такі групи засобів, що відображають:

- функції, які система повинна реалізувати;
- діяльності системи;
- відношення між даними;
- поведінку системи, що залежить від часу (аспекти реального часу).

Серед всієї множини засобів розв'язання даних задач в методологіях структурного аналізу найчастіше застосовуються такі [1,3]:

SADT (Structured Analysis and Design Technique) або її підмножина IDEF0, що є сукупністю методів, правил і процедур, призначених для побудови функціональної моделі об'єкта довільної предметної області. Модель SADT відображає функціональну структуру об'єкта, тобто його дії і зв'язок між ними;

IDEF3 (Icam DEFinition) — технологія для збирання описань процесів від експертів і для проектування моделей процесу, в якій відображається послідовність діяльностей і паралелізм;

DFD (Data Flow Diagrams) — діаґрами потоків даних спільно зі словниками даних та специфікаціями або мініспецифікаціями процесів;

STD (State Transition Diagrams) — діаґрами переходів станів;

ERD (Entity-Relationship Diagrams) — діаґрами "сутність-зв'язок".

Всі вони містять ґрафічні і текстові засоби моделювання: перші — для вигоди демонстрування основних компонентів моделі, другі — для забезпечення точного визначення її компонентів та зв'язків.

Логічна DFD:

- відображає зовнішні відносно системи джерела та витоки (адресати) даних;
- ідентифікує логічні функції (процеси) та групи елементів даних, що пов'язують одну функцію з іншою (потоки);
- ідентифікує сховища (накопичувачі) даних, до яких здійснюється доступ.

Структури потоків даних та визначення їхніх компонентів зберігаються і аналізуються в словникові даних. Кожна логічна функція (процес) може бути деталізована за допомогою DFD нижнього рівня; коли подальша деталізація перестає бути корисною, переходять до описання логіки функції за допомогою специфікації процесу (міні-специфікації). Вміст кожного сховища також зберігають в словникові даних, модель даних сховища розкривається за допомогою ERD. У випадку наявності застосувань реального часу DFD доповнюється засобами опису поведінки системи, що залежить від часу, які розкриваються за допомогою STD.

Перераховані засоби дають повний опис системи незалежно від того, чи вона вже існує, чи розробляється від початку. Таким чином будується логічна функціональна специфікація — докладний опис того, що повинна робити система, звільнений наскільки це

можливо від розгляду шляхів реалізації. Це дає проектувальнику чітке уявлення про остаточні результати, до яких слід і можна прагнути.

Нотації об'єктно-орієнтованого підходу

Хоча відомо багато різних нотацій для відображення основних понять ОП та відношень між ними, на практиці найбільшого поширення набули декілька [2,4,6]:

- **Object-Oriented Analysis** (OOA) Пітера Кода та Єда Юрдона (Coad and Yourdon),
- **Object-Oriented-Design** (OOD) Ґраді Буча (Grady Booch),
- **Object Modelling Technique** (OMT) Джіма Рамбо (James Rumbaugh),
- **Unified Modelling Language** (UML) – уніфікована мова моделювання, створена як узагальнення та спроба стандартизації наявних нотацій Ґраді Бучем, Джімом Рамбо та Айваром Якобсоном (Ivar Jacobson).

Нотація Ґраді Буча ґрунтується на діаґрамах класів (логічний аспект проекту), діаґрамах станів та переходів (динамічний аспект моделі окремого класу або системи загалом), діаґрамах об'єктів (існуючі об'єкти та їх зв'язки в логічному проекті системи), діаґрамах взаємодії (виконання сценарію, як і діаґрама об'єктів), діаґрамах модулів (розподіл класів та об'єктів за модулями у фізичному проекті системи), діаґрамах процесів (відображення розподілу процесів за процесорами в фізичному проекті системи).

Об'єктна модель в підході Пітера Кода та Едварда Юрдона (ООА) розглядається як сукупність таких компонентів:

- проблемна область (PD – Problem Domain), яка включає потрібні класи предметної області;
- взаємодія з користувачем (HI – Human Interaction), до складу якої входять вікна та звіти;
- управління даними (DM – Data Management), яка включає бази даних та об'єкти сервера;
- взаємодія систем (SI – System Interaction), яка містить інші системи;
- "не тепер" (NTT – Not This Time), яка включає те, що на даному етапі не розглядається, але потенційно може розглядатися на наступних ітераціях ООА.

Кожен з класів точно відповідає одній з компонент моделі. Компоненти моделі застосовуються для поділу класів на сенсовні підмножини. Компонента PD знає про інші компоненти мало. Вона розсилає повідомлення всім іншим компонентам з метою вирішення, як реаґувати на зміни. HI взаємодіє з PD з приводу всього того, що необхідно для взаємодії з людиною. DM взаємодіє з HI, коли в HI обираються дії "зберегти" та "завантажити". SI взаємодіє з PD щодо всього того, що необхідно для взаємодії систем.

Така орґанізація класів спрощує моделювання, особливо поточне моделювання всередині кожної з компонент моделі, сприяє повторному використанню (можливість застосувати ті ж класи проблемної області навіть при роботі із зовсім іншими класами взаємодії з користувачами. Компоненти моделі використовуються як план та керівництво при пошукові об'єктів.

UML – це мова моделювання, призначена для специфікації, візуалізації, конструювання і документування інформаційних систем. UML декларується як мова, що містить у собі в уніфікованому вигляді найкращі відомі практичні методи ґрафічного моделювання. Це є певним недоліком, оскільки таке зібрання еклектичне і не у всіх нотаціях, що застосовуються в UML, прослідковується "чистий" об'єктний підхід, особливо в тих, що описують динамічну поведінку та паралелізм системи. Так само з застереженнями слід сприймати твердження про універсальну придатність UML для моделювання бізнесу й інших непрограмних систем. UML успадковує техніки Booch, OMT і OOSE, застосовуючи й інші зображальні засоби, є стандартом мови, а не стандартом процесу.

UML – це результат розвитку концепцій моделювання, по яких був досягнутий консенсус у середовищі головних світових компаній-розробників об'єктно-орієнтованого програмного забезпечення, а саме:

- домовленість про семантику і нотації для різноманітних аспектів сучасного моделювання в лаконічній формі;
- визначення достатньої семантики для нових технологій: компонентного програмування, розподілених обчислень і т.д.;
- реалізація механізму розширення для нарощування майбутніх методів моделювання поверх UML;
- визначення достатньої семантики для організації репозиторія моделей.

UML – це мова моделювання, тобто засіб моделювання, котрим потрібно вміти користуватися, знати, коли і де застосовувати ту або іншу техніку моделювання, чітко уявляти собі порядок проектування і структуру створюваної моделі. На такі питання може дати відповіді методологія. Поза предметною областю UML знаходяться мови програмування, інструментальні засоби та модель процесу (UML не нав'язує свою модель процесу проектування, але при розробці мови автори вважали, що процес володіє наступними властивостями: керований прецедентами, ітеративний і інкрементний).

Графічна нотація – це відображення візуального представлення в семантику мови. Як згадувалося раніше, UML є результатом синтезу трьох технік моделювання, і, отже, успадковує їхню графічну нотацію з певними модифікаціями. Проект, створений засобами мови UML, може використовувати наступні типи діаграм (згруповані відповідно до їхнього призначення).

Діаграми, що відображають функціональні вимоги до системи та її статичну структуру:

- діаграма прецедентів (варіантів) використання (Use Case diagram);
- діаграма класів (Class diagram).

Діаграми, що дозволяють описувати аспекти поведінки системи:

- діаграма станів (Statechart diagram);
- діаграма активностей (Activity diagram).

Діаграми, що описують взаємодію між об'єктами системи:

- діаграма послідовності (Sequence diagram);
- діаграма кооперації (Collaboration diagram).

Діаграми, пов'язані з фізичною реалізацією системи:

- діаграма пакетів (компонент) – (Package diagram, Component diagram);
- діаграма розміщення (Deployment diagram).

Коротка характеристика призначення окремих діаграм наведена в таблиці 1.

Таблиця 1.

Діаграмні засоби UML

Засіб	Призначення
Діаграма варіантів (прецедентів) використання (Use Case Diagram)	Список усіх варіантів використання фактично визначає функціональні вимоги до інформаційної системи на ранніх етапах проекту, за допомогою яких може бути сформульоване технічне завдання, покликаний забезпечити гнучкий і ефективний механізм взаємодії між розробником і замовником проекту.
Діаграма класів (Class Diagram)	Діаграма класів відображає статичну структуру системи чи її частини, а саме статичну структуру понять, типів та класів. Поняття відображають уявлення користувачів про реальний світ, типи – інтерфейси компонентів ПЗ, класи – реалізацію компонентів ПЗ.

Діаґрама станів (Statechart diagram)	Діаґрама станів описує реакцію об'єкта на асинхронні зовнішні події, поведінку одного об'єкта в багатьох варіантах використання.
Діаґрама активностей (діяльностей) (Activity diagram)	Служить для описання операцій класів, коли необхідно представити алгоритм їх реалізації, при цьому кожний крок є виконанням операції деякого класу. Може відображати множину об'єктів в багатьох варіантах використання, в єдиному варіанті використання або реалізацію метода.
Діаґрама послідовності (Проходження) (Sequence diagram) – належить до діаґрам взаємодії (Interaction Diagram)	Діаґрама відображає динаміку взаємодії – послідовність передачі повідомлень у часі, порядок, вигляд і ім'я повідомлення; на діаґрамі зображаються винятково ті об'єкти, що безпосередньо беруть участь у взаємодії, і не показуються можливі статичні асоціації з іншими об'єктами.
Кооперативна діаґрама (Collaboration diagram) - належить до діаґрам взаємодії	Динаміка взаємодії ілюструється за допомогою нумерації подій, просторове розташування об'єктів відображає їх статичну взаємодію
Діаґрама пакетів (Package diagram) компонентів (Component diagram)	Служить для групування класів в об'єкти вищого рівня – пакети або компоненти; має в собі пакети класів та залежності між ними; по суті, форма діаґрами класів
Діаґрама розміщення (Deployment diagram)	Відображає фізичні взаємозв'язки між програмними та апаратними компонентами системи.

Інструментальні засоби проектування ІС

На сучасному ринку засобів розробки ІС досить багато систем, що реалізують структурну чи об'єктну методології проектування [1-3,5]. Відомими виробниками є RATIONAL SOFTWARE, PLATINUM TECHNOLOGY, що спеціалізуються в розробці CASE-систем, та ORACLE, яка має в своєму арсеналі практично весь спектр продуктів — від розроблення до експлуатації ІС, а також конкретні прикладні системи, що вирізняє її в сенсі комплексності серед конкурентів.

Для проведення аналізу і реорганізації бізнес-процесів PLATINUM TECHNOLOGY (продукти BPWin та ERWin придбані CA) пропонує CASE-засіб верхнього рівня BPWin, що підтримує методології IDEF0 (функціональна модель), IDEF3 (WorkFlow Diagram) і DFD (DataFlow Diagram). Якщо в процесі моделювання потрібно відобразити специфічні сторони технології підприємства, BPwin дозволяє перемкнутися на будь-якій гілці моделі на нотацію IDEF3 чи DFD і створити змішану модель. Нотація DFD включає такі поняття, як зовнішнє посилання і сховище даних, що робить її більш зручною (порівняно з IDEF0) для моделювання документообігу. Методологія IDEF3 включає елемент "перехрестя", що дозволяє описати логіку взаємодії компонентів системи.

На основі моделі BPwin можна побудувати модель даних. Для побудови моделі даних PLATINUM TECHNOLOGY пропонує ERwin. Хоча процес перетворення моделі Bpwin у модель даних погано формалізується і тому не цілком автоматизований, PLATINUM TECHNOLOGY пропонує зручний інструмент для полегшення побудови моделі даних на основі функціональної моделі – механізм двонаправленого зв'язку BPwin – Erwin. ERwin має два рівні представлення моделі – логічний і фізичний. На логічному рівні дані не пов'язані з конкретною СУБД, тому можуть бути наочно представлені навіть не для спеціалістів. Фізичний рівень даних – це, власне кажучи, відображення системного каталогу, що залежить від конкретної СУБД. ERwin дозволяє проводити процеси прямого і зворотного проектування БД. Це означає, що за моделлю даних можна зґенерувати схему БД чи автоматично створити модель даних на основі інформації системного каталогу.

Крім того, ERwin дозволяє кореґувати модель і вміст системного каталогу після редаґування одного або іншого. ERwin інтегрується з популярними засобами розробки клієнтської частини – (PowerBuilder, Visual Basic, Delphi) безпосередньо чи за допомогою продуктів сторонніх виробників, що дозволяє автоматично ґенерувати код прикладної програми, готовий до компіляції і виконання.

Фірма PLATINUM TECHNOLOGY пропонує систему організації спільної роботи Model Mart - сховище моделей, до якого відкритий доступ для учасників проекту створення ІС. Model Mart задовольняє усі вимоги, що висуваються до засобів розробки великих ІС, а саме:

• **Спільне моделювання.** Кожен учасник проекту має інструмент пошуку і доступу до моделі, яка його цікавить, в будь-який час. При спільній роботі використовуються три режими: незахищений, захищений і режим перегляду. У режимі перегляду забороняється будь-яка зміна моделей. У захищеному режимі модель, з якою працює один користувач, не може бути змінена іншими користувачами. У незахищеному режимі користувачі можуть працювати з загальними моделями в реальному масштабі часу.

• **Створення бібліотек рішень.** Model Mart дозволяє формувати бібліотеки стандартних рішень, що включають найвдаліші фрагменти реалізованих проектів, накопичувати і використовувати типові моделі, поєднуючи їх у разі необхідності “складання” великих систем.

• **Керування доступом.** Для кожного учасника проекту визначаються права доступу, відповідно до яких вони одержують можливість працювати тільки з визначеними моделями.

• **Архітектура Model Mart.** Model Mart реалізована в архітектурі клієнт - сервер. Як платформи реалізації сховища обрані СУБД Sybase, Microso/E SQL Server, Informix і Oracle. Клієнтськими прикладними програмами є ERwin і BPwin. У Model Mart реалізований доступ до сховища моделей через API, що дозволяє постійно нарощувати можливості інтегрованого середовища шляхом включення нових інструментів моделювання й аналізу.

ERwin забезпечує автоматичну ґенерацію коду клієнтської частини на основі моделі предметної області, а саме код ґенерується на основі моделі IDEF1X, тобто фактично на основі реляційної моделі даних, що безпосередньо не містить інформації про бізнес-процеси. Як наслідок цього зґенерований код не може забезпечити функціональність прикладної програми зі складною бізнесом-логікою.

CASE-засобами, що підтримують методологію об‘єктно-орієнтованого проектування, є Rational Rose (продукт RATIONAL SOFTWARE) та PLATINUM Paradigm Plus (продукт PLATINUM TECHNOLOGY). Ці інструменти дозволяють будувати об’єктні моделі в різних нотаціях (OMT, UML, Booch) і на основі отриманої моделі ґенерувати прикладні програми мовами програмування C++, Visual Basic, Java, Ada, Smalltalk і ін. Оскільки ґенерація коду реалізована на основі знань предметної області, а не на основі реляційної структури даних, отриманий код повніше відображає бізнес-логіку порівняно з ERwin. Rational Rose і Paradigm Plus підтримують не лише пряму ґенерацію коду, але і зворотне проектування, тобто створення об'єктної моделі за вихідним кодом прикладної програми. Своєрідним визнанням заслуг Rational Software стало включення спрощеного варіанта продукту Rational Rose у пакет розробника Microsoft Visual Studio (під назвою Visual Modeler).

Для ґенерації схеми БД об'єктну модель необхідно конвертувати в модель даних IDEF1X. Модуль ERwin Translation Wizard (PLATINUM TECHNOLOGY) дозволяє перевантажувати об’єктну модель Rational Rose у модель даних ERwin (і назад) і, за допомогою ERwin, зґенерувати схему БД на кожній з підтримуваних у ERwin СУБД. Для зв'язування об’єктної моделі, створеної в PLATINUM Paradigm Plus, з моделлю даних не потрібно додаткових утиліт.

Rational Rose здатний ґенерувати описи мовами Interface Definition Language (IDL) для застосувань CORBA і Data Definition Language (DDL) для застосувань доступу до баз даних, у тому числі і Oracle 8.

Фірма TogetherSoft LLC випускає три варіанти свого пакета, що ґрунтується на UML: Together/J Developer для розробників, що використовують Java, Together/C++ Developer для проектів на C++ і об'єднану версію для обох мов — Together/E (Enterprise).

Слід зазначити ще одну тенденцію, що полягає в переорієнтації інструментальних засобів, спочатку створених для підтримки структурних методів розробки, на методи ОП. Комбіновані CASE підтримують як ОП, так і структурні методології.

Продукт фірми ORACLE – Oracle Designer – забезпечує підтримання всіх етапів циклу життя інформаційної системи, починаючи з найзагальніших описів предметної області і до отримання та супроводу готового проґрамного продукту. Для описання моделі орґанізації, оптимізації структури підрозділів, моделювання та реінженерії бізнес-процесів використовується Business Process Modeller – інструмент, що входить до складу Oracle Designer. Oracle Designer дозволяє будувати концептуальні моделі двох типів – інформаційні (відображають існуючі інформаційні структури та взаємні зв'язки між ними) та функціональні (описують технологію та способи опрацювання інформації, що використовуються в тій чи іншій області). Ці моделі є ефективним засобом комунікації між проектувальниками та користувачами в процесі формулювання та уточнення задач.

Такий підхід, при якому специфікації компонент інформаційної системи відділені від їх конкретної реалізації, дозволяє внести в моделі довільні зміни в бізнес-процесах та зґенерувати модифіковане застосування, що ґрунтується на нових схемах ведення бізнесу, зі збереженням всього того, що було розроблено раніше. Окрім того, Oracle Designer автоматично створює звіти, в яких наявна вся інформація про проект та його поточний стан.

Проблеми застосування CASE-засобів в аналізі систем та реінженерії бізнес-процесів

Сучасний ринок диктує жорсткі вимоги до ІС, що проектуються та реалізуються. Системи повинні бути надійними, високопродуктивними, гнучкими стосовно будь-яких змін у вимогах до них, забезпечувати можливість масштабування і нарощування функціональності. Крім того, розробка повинна вестися швидко, якісно і з мінімальними витратами.

Розробка моделі (а точніше – комплексу моделей) складної системи перед безпосередньою реалізацією інформаційної системи є невід'ємною частиною всього проекту. Якісна модель є основою для безперешкодної взаємодії в команді розробників і ґарантує загальний успіх розпочатого проекту. Побудова моделей є необхідною, оскільки "з першого погляду" охопити як усю систему загалом, так і окремі її функціональні частини неможливо. Із зростанням складності систем, що проектуються, усе більше виявляється необхідність у наявності зручних засобів моделювання.

Етап аналізу та формулювання вимог є одним з найважливіших та найскладніших в сучасному проектуванні ІС, оскільки на цьому етапі необхідно зрозуміти, що пропонується реалізувати, та задокументувати отримані результати (коли вимоги не зафіксовані та не є доступними для учасників проекту, то вони ніби й не існують). Мова, якою формулюються вимоги, повинна бути достатньо простою та зрозумілою замовнику. З цим пов'язаний такий комплекс проблем:

- аналітику складно отримати докладну інформацію для оцінки вимог до системи з точки зору замовника;
- замовник не має достатньої інформації про проблеми, що виникають при опрацюванні даних, щоб знати, що може бути виконаним, а що – ні;
- специфікація системи в багатьох випадках незрозуміла для замовника внаслідок об'єму та специфічних технічних термінів;

- у випадку, коли специфікація зрозуміла для замовника, вона буде недостатньою для проектантів та програмістів, що створюють систему.

Звичайно, не слід вважати, що застосування лише об'єктно-орієнтованих засобів розв'яже всі проблеми проектування, тим більше, що існують усталені методології реінженерії бізнес-процесів й інші подібні. Врешті-решт якість інформаційної системи визначається не застосуванням ультрамодних засобів розробки, а ступенем задоволення потреб замовників та рівнем кваліфікації розробників – за допомогою найсучасніших засобів і недостатньої кваліфікації можна лише надзвичайно швидко спроектувати неякісну систему, а стратегічні прорахунки виправдовувати ітераційним характером об'єктно-орієнтованого процесу проектування.

Так, об'єктно-орієнтована методологія визнана сьогодні базовою методологією проектування ІС, і на це є дуже вагомі причини, але – застосування її не є універсальним і не повинно привести до зникнення всього іншого – врешті-решт користувач визначає якість тієї чи іншої ІС, і опосередковано – область застосування тієї чи іншої методології чи технології проектування ІС. Натомість твердження про те, що об'єктно-орієнтовані CASE-засоби є базовими, наприклад, в BPR чи аналізі систем, є перебільшенням, і в основному зустрічаються в рекламних проспектах чи на сайтах власне виробників цих засобів.

Цьому сприяє наявність слова "аналіз" в назві такого важливого етапу, як об'єктно-орієнтований аналіз, і свідоме чи несвідоме прагнення ототожнити його з системно-аналітичними дослідженнями, що виконуються як початковий етап розробки інформаційної системи чи реінженерії бізнес-процесів. Ці дослідження далеко не тотожні об'єктно-орієнтованому аналізу, прийоми якого в свою чергу можуть бути успішно використані й під час таких системно-аналітичних досліджень, що створює ілюзію рівнозначимості цих досліджень та можливості заміни об'єктно-орієнтованим аналізом системно-аналітичних досліджень. Провідні консалтингові фірми використовують свої методології для розв'язання проблеми ефективної комунікації замовник-виконавець, виявлення проблемних ситуацій та формулювання вимог, не відмовляючись від інтеграції перспективних технологій інших фірм.

В аналізі системи та проведенні реінженерії беруть участь фахівці двох типів - професіонали в області бізнесу, що підлягає реінженерії, і проектувальники-розробники інформаційних систем. Досвід реінженерії показує, що по-справжньому успішне і новаторське впровадження інформаційних технологій є унікальним творчим процесом: керуючі компаній і фахівці-технологи, знайомлячись з методами ІТ, самі роблять відкриття щодо можливостей їхнього використання у своєму конкретному бізнесі. У той же час створення високоякісних інформаційних систем вимагає участі професіоналів в області ІТ. Виникає проблема пошуку спільної мови, що покликана забезпечити ефективну комунікацію між замовниками та фахівцями в області ІС та сучасних інформаційних технологій. Призначення графічних моделей – полегшити спілкування з експертами орґанізації – але в цьому випадку використовується не лише та чи інша графічна нотація як зручний засіб комунікації, а й весь інструментарій системного аналітика для отримання загального уявлення про систему (який є ніби поза кадром, але вирішальним чином впливає на якість аналізу та подальшої розробки).

Деякі з пропозицій зводяться до інтеграції сучасних технологій моделювання і розробки складних систем: об'єктно-орієнтованих методів, CASE-технологій, інженерії знань, імітаційного моделювання процесів і методів швидкої розробки застосувань RAD (Rapid Application Development), інші зосереджуються на засобах класичної структурної методології проектування ІС. Останнім часом панацеєю проголошується UML та RUP (Rational Unified Prozess), хоча зрозуміло, що такий засіб аналізу, як діаґрами прецедентів використаня (Use Case Diagram) не дозволяють повною мірою моделювати бізнес-процеси,

хоча їх користь в процесі уточнення вимог є безсумнівною. Разом з тим такий засіб, як ORACLE Designer, що ґрунтується на структурній методології, широко використовується в практиці проектування ІС власне внаслідок охоплення повного циклу життя системи.

Існує доволі широкий спектр думок щодо можливостей використання об'єктно-орієнтованих CASE-засобів на початкових етапах проектування – від цілковитого неприйняття до інтеґрації їх в методології-оболонки [8]. Так, керівник компанії IDS Scheer Аугуст Вільґельм Шеєр наголошує на різниці між інструментарієм та методологією ARIS, оскільки, на його думку, підтримку нових методів можна включати в інструментарій у міру того, як такі методи з'являються.

Таким чином, існує "металар" моделювання системи ("ділове", бізнес-моделювання) у вигляді певних методологій, як EPC (event-driven process chain), інтеґрованих ланцюжків поставок (Supply Chain), методології Btree Networks, IDT та інших, різниця між якими є суто формальною – всі вони служать для перетворення знань про підприємство в інформаційні технології.

На підтвердження цієї тези наводиться приклад реалізації UML шляхом "вбудовування " його в методологію ARIS, що було виконане за декілька тижнів. Реальна цінність методу в цьому випадку визначається тим, які складові бізнес-процесу він здатний описати. Це означає, що питання застосування того чи іншого методу є практичним питанням, і єдиним інтеґруючим вважається підхід з точки зору процесів. В той же час існують методології, що ґрунтуються на функціональній декомпозиції, і небезпідставно пропонується використання нотації IDEF0.

Висновки

1. На попередніх етапах проектування інформаційних систем – системноаналітичних досліджень та формулювання вимог доцільно використовувати методології, що дозволяють цілісно змоделювати цілі, функції, процеси та поведінку системи, а саме – методології бізнес-моделювання, що є достатньо гнучкими і дозволяють включати до свого інструментарію за необхідності підтримку нових методів.

2. Основними проблемами початкових етапів проектування є проблема комунікації замовник-виконавець та проблема трансляції – перетворення отриманих шляхом комунікації знань про підприємство в інформаційні технології.

3. Проблема комунікації зовнішньо розв'язується через використання тих чи інших нотацій – діаґрамних технік, які не є самоцінними і скеровані на продуктивну реалізацію основного – отримання необхідних знань, що можливо лише за умови високої кваліфікації системного аналітика.

4. Розвиток методології та засобів об'єктного підходу не означає зникнення структурного підходу, оскільки отримують знання в більшості випадків шляхом застосування структурної методології та відповідних нотацій, а трансляція в об'єктне представлення на етапі ООА можлива не лише на рівні методів, але й засобів.

1. Калянов Г.Н. CASE. Структурный системный анализ. «Лори», М., 1996. 2. Ларман К. Применение UML и шаблонов проектирования. Введение в объектно-ориентированный анализ и проектирование. М., 2001. 3. Маклаков С.В. BPwin и Erwin. CASE-средства разработки информационных систем., М., 1999. 4. Эдвард Йордон, Карл Аргила. Структурные модели в объектно-ориентированном анализе и проектировании., М., 1999. 5. Колетски П., Дорси П. ORACLE Designer. Настольная книга пользователя. «Лори», М., 1999. 6. Фаулер М., Скотт К. UML в кратком изложении., М., 1999. 7. Коад П., Норт Д., Мейфилд М. Объектные модели. Стратегии, шаблоны и приложения., М., 1999. 8. Шеер А.В. Практика моделирования – лучший критерий// Computerworld Россия, Директору информационной службы, Моделирование и CASE-средства. – 2000. 11. С.10 – 13.