**Лабораторна робота №10.**  **Застосування методології RAD Model для інкрементній моделі ЖЦ ПЗ**

**Мета роботи**: засвоєння навичок визначення

**Завдання:**

1. Ознайомитися з теоретичною частиною.
2. Перегляньте сайт Української залізниці https://www.uz.gov.ua/ .
3. Для реалізації аналогічного проекту (гіпотетичної залізниці) за методологією RAD визначте:
4. перелік основних функціональних складових проекту та загальну інформаційну модель проекту;
5. які з основних функціональних складових проекту можуть виконуватися паралельно;
6. на які інкременти можна розкласти розробку цього проекту.
7. напрями спеціалізації ІТ – фахівців, необхідних для розроблення цього проекту;
8. які сутності/об’єкти можна визначити для цього проекту та які з цих об’єктів можна використати в інших проектах.
9. По закінченню лабораторну роботу потрібно здати на перевірку викладачеві, надіславши електронною поштою на адресу [t.i.lumpova@gmail.com](mailto:t.i.lumpova@gmail.com). Якщо викладач знаходить помилки чи неточності, він може повернути роботу на доопрацювання.

Файл з роботою повинен мати назву в такому форматі:

**KPZ <Номер групи><Номер лекції / лабораторної> [літера позначення типу роботи L – лекція, R – лабораторна]<Прізвище англійською>**.. Наприклад, **KPZ4101R**buts.doc.

Не копіюйте фрагментів з різних інформаційних джерел, подумайте і викладіть свою точку зору. При наявності робіт-"близнюків" відповідь буде зараховуватися першому за часом надсилання.

Тему в заголовку листа записати

**KPZ<Номер групи>-ЛР<Номер лабораторної>-<Прізвище >**

**Строк виконання цієї роботи ІПЗ-41 –**

**ІПЗ-42 – 06.11.2021**

Всі запитання, що виникнуть, надсилайте на електронну адресу викладача, Тему в заголовку листа записати

**KPZ <Номер групи>-Запитання-<Прізвище >**.

**КОНТРОЛЬНІ ПИТАННЯ:**

1. Дайте визначення інкрементній моделі ЖЦ ПЗ.
2. В чому буде полягати прототипування за інкрементною моделлю?
3. В чому полягають особливості спіральної моделі?
4. Які переваги надає залучення користувачів до розробки?

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

**RAD** (Rapid Application Development) - методологія швидкої розробки додатків, що передбачає застосування інструментальних засобів візуального моделювання (прототипування) і розробки. RAD передбачає невеликі команди розробки, терміни до 4 місяців й активне залучення замовника з ранніх етапів. Дана методологія спирається на вимоги, але також існує можливість їхніх змін під час розробки системи. Такий підхід дозволяє скоротити витрати і звести час розробки до мінімуму.

Під терміном **RAD** зазвичай розуміється процес розробки ПЗ, що містить 3 елементи:

* невелику команду програмістів (від 2 до 10 осіб);
* короткий, але ретельно пророблений виробничий графік (від 2 до 6 міс.);
* повторюваний цикл, при якому розробники по мірі набуття форми додатком запитують і реалізують у продукті вимоги, отримані через взаємодію з замовником.

Команда розробників повинна представляти із себе групу професіоналів, які мають досвід в аналізі, проектуванні, генерації коду і тестуванні ПЗ з використанням CASE-засобів. Члени колективу повинні також вміти трансформувати в робочі прототипи пропозиції кінцевих користувачів.

## Основні принципи методології RAD.

Основні принципи методології RAD можна звести до таких:

* Використовується ітераційна (спіральна) модель розробки;
* Повне завершення робіт на кожному з етапів життєвого циклу не обов’язкове;
* В процесі розробки інформаційної системи необхідна тісна взаємодія із замовником і майбутніми користувачами;
* Необхідне використання CASE-засобів і засобів швидкої розробки додатків;
* Необхідне використання засобів управління конфігурацією, що полегшують внесення змін до проекту і супровід готової системи;
* Необхідне використання прототипів, що дозволяє повніше з'ясувати і реалізувати потреби кінцевого користувача;
* Тестування і розвиток проекту здійснюються одночасно з розробкою;
* Розробка ведеться нечисленною і добре керованою командою професіоналів;
* Необхідні грамотне керівництво розробкою системи, чітке планування і контроль виконання робіт.

Засоби RAD дали можливість реалізовувати зовсім іншу в порівнянні з традиційною технологію створення додатків. Інформаційні об'єкти формуються як якісь діючі моделі (прототипи), чиє функціонування узгоджується з користувачем, а потім розробник може переходити безпосередньо до формування закінчених додатків, не втрачаючи з уваги загальної картини проектованої системи.

Можливість використання подібного підходу в значній мірі є результатом застосування принципів об'єктно-орієнтованого проектування, Застосування об'єктно-орієнтованих методів дозволяє подолати одну з головних труднощів, що виникають при розробці складних систем – колосальний розрив між реальним світом (предметною областю описуваної проблеми) і імітаційним середовищем.

Використання об'єктно-орієнтованих методів дозволяє створити опис (модель) предметної області у вигляді сукупності об'єктів – сутностей, поєднуючих дані і методи обробки цих даних (процедури). Кожен об'єкт володіє своєю власною поведінкою і моделює деякий об'єкт реального світу. З цієї точки зору об'єкт є цілком відчутною річчю, яка демонструє певну поведінку.

В об'єктному підході акцент переноситься на конкретні характеристики фізичної або абстрактної системи, що є предметом програмного моделювання. Об'єкти є цілісністю, яка не може бути порушена. Таким чином, властивості, що характеризують об'єкт і його поведінку, залишаються незмінними. Об'єкт може тільки міняти стан, управлятися або ставати в певне відношення до інших об'єктів.

Широку популярність об'єктно-орієнтоване програмування отримало з появою візуальних засобів проектування, коли було забезпечено злиття (інкапсуляція) даних з процедурами, що описують поведінку реальних об'єктів, в об'єкти програм, які можуть бути відображені певним чином в графічному середовищі користувача. Це дозволило приступити до створення програмних систем, максимально схожих на реальні, і досягати найвищого рівня абстракції. У свою чергу, об'єктно-орієнтоване програмування дозволяє створювати більш надійні коди, оскільки у об'єктів програм існує точно визначений і жорстко контрольований інтерфейс.

При розробці додатків за допомогою інструментів RAD використовується множина готових об'єктів, які зберігаються в загальнодоступному сховищі. Однак забезпечується і можливість розробки нових об'єктів. При цьому нові об'єкти можуть розроблятися як на основі існуючих, так і «з нуля».

Інструментальні засоби RAD мають зручний графічний інтерфейс користувача і дозволяють на основі стандартних об'єктів формувати прості програми без написання коду програми. Це є великою перевагою RAD, оскільки в значній мірі скорочує рутинну роботу з розробки інтерфейсів користувача (при використанні звичайних засобів розробка інтерфейсів є досить трудомісткою задачею, що забирає багато часу). Висока швидкість розробки інтерфейсної частини додатків дозволяє швидко створювати прототипи і спрощує взаємодію з кінцевими користувачами.

Таким чином, інструменти RAD дозволяють розробникам сконцентрувати зусилля на сутності реальних ділових процесів підприємства, для якого створюється інформаційна система. У результаті це приводить до підвищення якості розроблюваної системи.

Застосування принципів об'єктно-орієнтованого програмування дозволило створити принципово нові засоби проектування додатків, які були названі засобами візуального програмування. Візуальні інструменти RAD дозволяють створювати складні графічні інтерфейси користувача взагалі без написання коду програми. При цьому розробник може на будь-якому етапі спостерігати, що закладається в основу прийнятих рішень.

Візуальні засоби розробки оперують в першу чергу зі стандартними інтерфейсними об'єктами – вікнами, списками, текстами, які легко можна зв'язати з даними з бази даних і відобразити на екрані монітора. Інша група об'єктів являє собою стандартні елементи управління – кнопки, перемикачі, прапорці, меню тощо, за допомогою яких здійснюється управління відображеними даними. Всі ці об'єкти можуть бути стандартним чином описані засобами мови, а описи збережені для подальшого повторного використання.

В даний час існує досить багато різних візуальних засобів розробки додатків. Але всі вони можуть бути розділені на дві групи – універсальні і спеціалізовані.

Серед універсальних систем візуального програмування зараз найбільш поширені такі, як С++, Visual Studio. Універсальними ми їх називаємо тому, що вони не орієнтовані на розробку лише додатків баз даних – з їх допомогою можуть бути розроблені додатки майже будь-якого типу, у тому числі й інформаційні програми. Причому програми, які розроблюються за допомогою універсальних систем, можуть взаємодіяти практично з будь-якими системами управління базами даних. Це забезпечується як використанням драйверів ODBC або OLE DB, так і застосуванням спеціалізованих засобів (компонентів).

Спеціалізовані засоби розробки орієнтовані тільки на створення додатків баз даних. Причому, як правило, вони прив'язані до конкретних систем управління базами даних. Як приклад таких систем можна привести Power Builder фірми Sybase (природно, призначений для роботи з СУБД Sybase Anywhere Server) і Visual FoxPro фірми Microsoft.

Оскільки завдання створення прототипів і розробки користувальницького інтерфейсу, по суті, злилися, програміст отримав безперервний зворотний зв'язок з кінцевими користувачами, які можуть не тільки спостерігати за створенням додатків, але й активно брати участь у ньому, коригувати результати і свої вимоги. Це також сприяє скороченню термінів розробки і є важливим психологічним аспектом, який привертає до RAD все більшу кількість користувачів.

## Життєвий цикл ПЗ відповідно RAD

Життєвий цикл ПЗ за методологією RAD складається з чотирьох фаз:

* фаза аналізу і планування вимог;
* фаза проектування;
* фаза побудови;
* фаза впровадження.

### Фаза аналізу.

На фазі аналізу і планування вимог користувачі системи визначають функції, які вона повинна виконувати, виділяють найбільш пріоритетні з них, які потребують опрацювання в першу чергу, описують інформаційні потреби. Визначення вимог виконується в основному силами користувачів під керівництвом фахівців-розробників. Обмежується масштаб проекту, визначаються часові рамки для кожної з наступних фаз. Крім того, визначається сама можливість реалізації даного проекту у встановлених рамках фінансування, на даних апаратних засобах і т. п. Результатом даної фази повинні бути список і пріоритетність функцій майбутньої ІС, попередні функціональні та інформаційні моделі ІС.

### Фаза проектування.

На фазі проектування частина користувачів бере участь в технічному проектуванні системи під керівництвом фахівців-розробників. CASE-засоби використовуються для швидкого отримання працюючих прототипів додатків. Користувачі, які безпосередньо взаємодіють з ними, уточнюють і доповнюють вимоги до системи, що не були виявлені на попередній фазі. Більш докладно розглядаються процеси системи. Аналізується і, за необхідності, коригується функціональна модель. Кожен процес розглядається детально. За необхідності для кожного елементарного процесу створюється частковий прототип: екран, діалог, звіт, що усуває неясності або неоднозначності. Визначаються вимоги розмежування доступу до даних. На цій же фазі відбувається визначення набору необхідної документації.

Після детального визначення складу процесів оцінюється кількість функціональних елементів розроблюваної системи і приймається рішення про поділ ІС на підсистеми, які піддаються реалізації однією командою розробників за прийнятний для RAD-проектів час – близько 60 – 90 днів. З використанням CASE-засобів проект розподіляється між різними командами (ділиться функціональна модель). Результатом даної фази повинні бути:

* загальна інформаційна модель системи;
* функціональні моделі системи в цілому і підсистем, що реалізуються окремими командами розробників;
* точно визначені за допомогою CASE-засобу інтерфейси між автономно розробленими підсистемами;
* побудовані прототипи екранів, звітів, діалогів.

Всі моделі і прототипи повинні бути отримані із використанням тих CASE-засобів, які будуть використовуватися в подальшому при побудові системи. Дана вимога викликана тим, що в традиційному підході при передачі інформації про проект з етапу на етап може відбутися фактично неконтрольоване спотворення даних. Застосування єдиного середовища зберігання інформації про проект дозволяє уникнути цієї небезпеки.

На відміну від традиційного підходу, при якому використовувалися специфічні засоби прототипування, не призначені для побудови реальних додатків, а прототипи викидалися після того, як виконували завдання усунення нез’ясованих частин у проекті, в підході RAD кожен прототип розвивається в частину майбутньої системи. Таким чином, на наступну фазу передається більш повна і корисна інформація.

### Фаза впровадження.

На фазі впровадження проводиться навчання користувачів, організаційні зміни і паралельно з впровадженням нової системи здійснюється робота з існуючою системою (до повного впровадження нової). Так як фаза побудови досить нетривала, планування і підготовка до впровадження повинні починатися заздалегідь, як правило, на етапі проектування системи.

Наведена схема розробки ІС не є абсолютною. Можливі різні варіанти, які залежать, наприклад, від початкових умов, в яких ведеться розробка: розробляється зовсім нова система; вже було проведено обстеження підприємства і існує модель його діяльності; на підприємстві вже існує деяка ІС, яка може бути використана в якості початкового прототипу або повинна бути інтегрована з розробляється.

## Оцінка розміру додатків

Оцінка розміру додатків виконується на основі так званих функціональних елементів (екрани, повідомлення, звіти, файли і т.п.). Подібна метрика не залежить від мови програмування, на якій ведеться розробка. Розмір програми, яка може бути виконана за методологією RAD, для добре налагодженого середовища розробки ІС з максимальним повторним використанням програмних компонентів, визначається таким чином:

* <1000 функціональних елементів – одна людина;
* 1000-4000 функціональних елементів – одна команда розробників;
* > 4000 функціональних елементів – 4000 функціональних елементів на одну команду розробників.

## Логіка додатків, побудованих за допомогою RAD

Логіка додатка, побудованого за допомогою RAD є подійно-орієнтованою. Це означає наступне: кожен об'єкт, що входить до складу додатка, може генерувати події і реагувати на події, що генеруються іншими об'єктами. Прикладами подій можуть бути: відкриття і закриття вікон, натискання кнопки, натискання клавіші клавіатури, рух миші, зміна даних в базі даних і т. п.

Розробник реалізує логіку програми шляхом визначення обробника кожної події – процедури, виконуваної об'єктом при настанні відповідної події. Наприклад, обробник події «натискання кнопки» може відкрити діалогове вікно. Таким чином, управління об'єктами здійснюється за допомогою подій.

Обробники подій, пов'язаних з управлінням базою даних (DELETE, INSERT, UPDATE), можуть реалізовуватися у вигляді тригерів на клієнтському або серверному вузлі. Такі обробники дозволяють забезпечити посилальну цілісність бази даних при операціях видалення, вставки і оновлення, а також автоматичну генерацію первинних ключів.

## Застосування RAD.

Застосування технології RAD доцільно, коли:

* потрібне виконання проекту у стислі терміни (90 днів). Швидке виконання проекту дозволяє створити систему, що відповідає вимогам сьогоднішнього дня. Якщо система проектується довго, то вельми висока ймовірність, що за цей час істотно зміняться фундаментальні положення, що регламентують діяльність організації, тобто, система морально застаріє ще до завершення її проектування
* нечітко визначені вимоги до ПЗ. У більшості випадків Замовник дуже приблизно уявляє собі роботу майбутнього програмного продукту і не може чітко сформулювати всі вимоги до ПЗ. Вимоги можуть бути взагалі не визначені до початку проекту або можуть змінюватися по ходу його виконання
* проект виконується в умовах обмеженості бюджету. Розробка ведеться невеликими RAD групами у короткі терміни, що забезпечує мінімум трудовитрат і дозволяє вписатися у бюджетні обмеження
* інтерфейс користувача (GUI) є головним фактором. Немає сенсу примушувати користувача малювати картинки. RAD технологія дає можливість продемонструвати інтерфейс в прототипі, причому досить скоро після початку проекту
* проект великий, але піддається поділу на більш дрібні функціональні компоненти. Якщо передбачувана система велика, необхідно, щоб її можна було розбити на дрібні частини, кожна з яких має чітку функціональність. Вони можуть випускатися послідовно або паралельно (в останньому випадку залучається кілька RAD груп)
* ПЗ не володіє великою обчислювальною складністю

Слід, однак, відзначити, що методологія RAD, як і будь-яка інша, не може претендувати на універсальність, вона хороша в першу чергу для відносно невеликих проектів, що розробляються для конкретного замовника. Якщо ж розробляється типова система, яка не є закінченим продуктом, а являє собою комплекс типових компонент, централізовано супроводжуваних, які адаптуються до програмно-технічних платформ, СУБД, засобів телекомунікації, організаційно-економічних особливостей об'єктів впровадження і інтегрованих з існуючими розробками, на перший план виступають такі показники проекту, як керованість і якість, які можуть увійти в суперечність з простотою і швидкістю розробки. Для таких проектів необхідні високий рівень планування і жорстка дисципліна проектування, суворе дотримання заздалегідь розробленим протоколам і інтерфейсам, що знижує швидкість розробки.

Методологія RAD непридатна для побудови складних розрахункових програм, операційних систем або програм управління космічними кораблями, тобто програм, що вимагають написання великого обсягу (сотні тисяч рядків) унікального коду.

Не підходять для розробки за методологією RAD додатки, в яких відсутня яскраво виражена інтерфейсна частина, наочно визначає логіку роботи системи (наприклад, додатки реального часу) і додатки, від яких залежить безпека людей (наприклад, керування літаком або атомною електростанцією), так як ітеративний підхід передбачає, що перші кілька версій напевно не будуть повністю працездатні, що в даному випадку виключається.

Застосування методології RAD найбільш ефективно при створенні порівняно невеликих систем, що розробляються для конкретного замовника.

Як підсумок перерахуємо основні принципи методології RAD:

* розробка додатків ітераціями;
* необов'язковість повного завершення робіт на кожному з етапів життєвого циклу;
* обов'язкове залучення користувачів до процесу розробки ІС;
* необхідне застосування CASE-засобів, що забезпечують цілісність проекту;
* застосування засобів управління конфігурацією, що полегшують внесення змін до проекту і супровід готової системи;
* необхідне використання генераторів коду;
* використання прототипування, що дозволяє повніше з'ясувати і задовольнити потреби кінцевого користувача;
* тестування і розвиток проекту, здійснювані одночасно з розробкою;
* ведення розробки нечисленної добре керованої командою професіоналів;
* грамотне керівництво розробкою системи, чітке планування і контроль виконання робіт.

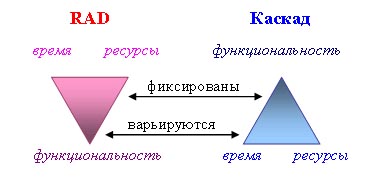
RAD припускає, що розробка програмного забезпечення (ПЗ) здійснюється невеликою командою розробників за термін близько трьох-чотирьох місяців шляхом використання інкрементного прототипування із застосуванням інструментальних засобів візуального моделювання та розробки. Технологія RAD передбачає активне залучення Замовника вже на ранніх стадіях – обстеження організації, вироблення вимог до системи. Причини популярності RAD випливають з переваг, які забезпечує ця технологія.

Найбільш істотними з них є:

* Висока швидкість розробки;
* Низька вартість;
* Висока якість.

Остання із зазначених властивостей має на увазі повне виконання вимог Замовника – як функціональних, так і нефункціональних, з урахуванням їх можливих змін у період розробки системи, а також отримання якісної документації, що забезпечує зручність експлуатації та супроводу системи. Це означає, що додаткові витрати на супровід відразу після поставки будуть значно меншими. Таким чином, повний час від початку розробки до отримання прийнятного продукту при використанні цього методу значно скорочується.

Технологію RAD доцільно застосовувати, коли чітко визначені деякі пріоритетні напрямки розробки проекту.



Порівняння RAD і Каскадного методу

1. *Необхідно виконання проекту в стислі терміни.* Швидке виконання проекту дозволяє створити систему, що відповідає вимогам сьогоднішнього дня. Якщо система проектується довго, то вельми висока ймовірність, що за цей час суттєво зміняться фундаментальні положення, що регламентують діяльність організації, тобто, система морально застаріє ще до завершення її проектування.
2. *Нечітко визначені вимоги до ПЗ.* В більшості випадків замовник дуже приблизно уявляє собі роботу майбутнього програмного продукту і не може чітко сформулювати всі вимоги до ПЗ. Вимоги можуть бути взагалі не визначені до початку проекту або можуть змінюватися по ходу його виконання.
3. *Проект виконується в умовах обмеженості бюджету.* Розробка ведеться невеликими RAD-групами в короткі терміни, що забезпечує мінімум трудовитрат і дозволяє вписатися в бюджетні обмеження.
4. *Інтерфейс користувача (GUI) є головний фактор.* Немає сенсу змушувати користувача малювати картинки. RAD-технологія дає можливість продемонструвати інтерфейс в прототипі, причому досить скоро після початку проекту.
5. *Можливо розбивка проекту на функціональні компоненти.* Якщо передбачувана система велика, необхідно, щоб її можна було розбити на дрібні частини, кожна з яких володіє чіткою функціональністю. Вони можуть випускатися послідовно або паралельно (в останньому випадку залучається кілька RAD-груп).
6. *Низька обчислювальна складність ПЗ.*

RAD-технологія не є універсальною, тобто її застосування доцільно не завжди. Наприклад, у проектах, де вимоги до програмного продукту чітко визначені і не повинні мінятися, залучення замовника в процес розробки не потрібно і більш ефективною може бути ієрархічна розробка ( каскадний метод). Те ж стосується проектів, ПЗ, складність яких визначається необхідністю реалізації складних алгоритмів, а роль і обсяг користувальницького інтерфейсу невеликий.