від частоти для досліджуваного кабелю мало відрізняються між собою та не перевищують максимально допустимих значень для кабелю UTP за стандартом TSB-67.

Висновки

- проведено моделювання поширення сигналу в середовищі типу "вита пара" на основі побудованої математичної моделі на частотах до 125 МГц;
- отримано формули для розрахунку коефіцієнта передачі за потужністю $\underline{K_p}(\omega)$ (3)-(18) та коефіцієнта затухання $\underline{\alpha_p}(\omega)$ (19)-(24) для середовища типу "вита пара";
- запропоновано нові методи розрахунку опору втрат і провідності втрат досліджуваного середовища (27), (28), при цьому при обчисленні опору втрат враховано опір лінії постійному струму, а також експериментально визначено значення коефіцієнта для середовища типу "вита пара";
- показано, що ця модель забезпечує значний збіг між результатами теоретичних та експериментальних досліджень;
- доцільно використати одержані результати при створенні математичних моделей телекомунікаційних каналів зв'язку.

1. Грицьків Р.Д. Основи теорії довгих ліній. – К.: Вища школа, 1974. – 144 с.
2. Лосев А.К. Теория линейных электрических цепей: Учеб. для вузов. – М.: Высш. шк., 1987. – 512 с.: ил.
3. Мансуров Н.Н., Попов В.С. Теоретическая электротехника. – М.: Госэнергоиздат, 1958. – 608 с.
4. Тимченко О.В., Горбатий І.В. Математична модель середовища типу "вита пара" // Збірник наукових праць ІПМЕ НАН України. – К. – 2002. – Вип. 16. – С. 147 – 157.

В. Заяць, Д. Іванов
Національний університет "Львівська політехніка"

УДК 628.321

ПРОЕКТ СИСТЕМИ РОЗПІЗНАВАННЯ РУКОПИСНОГО ТЕКСТУ

Описано проект побудови системи розпізнавання рукописного тексту. Наведено приклад розбиття системи на підсистеми із виділенням функціонального призначення кожної із них та вказано напрямки їх використання.

The article description project system of recognizing handwritten text. There is example of splitting system on subsystems (with definition roles and application each of them) in this article.

Вступ

Описано проект системи, призначеної для розпізнавання окремих літер рукописного тексту української абетки, а також для розпізнавання слів, що складаються з окремих літер української абетки. Система передбачає роботу з такими стандартними частинами проектування і програмування, як: система бази даних, система доступу до сховища даних, система сутностей програми, система відображення даних (інтерфейс користувача на стаціонарних комп'ютерних системах, мобільних пристроях, кишенькових комп'ютерах; файл; друкарка та ін.). Всі частини так чи інакше належать до одного з чотирьох етапів роботи з даними: зберігання, зчитування, підготовка до відображення та відображення. Наявність великої кількості реалізацій таких підчастин програми надала проектувальникам програмних засобів можливість відокремити певні стандарти щодо їх реалізації (так звані шаблони проектування). Наявність шаблонів дозволяє повторне використання вже написаного коду, масштабування коду, легкого його супроводу та підтримки, документування, тестування та верифікації. Саме на основі таких шаблонів проектування і буде вестися розробка системи розпізнавання рукописного тексту.

Опис проблеми

Незважаючи на те, що після виходу в світ першої роботи по розпізнаванню [1] пройшло майже півстоліття, з'являється цілий ряд нових робіт [2], пов'язаних з цією проблемою. Проблема розпізнавання є актуальною, оскільки потребує системного підходу до свого вирішення. Традиційно в силу тих чи інших причин вирішення проблеми розпізнавання об'єктів зводилося до формулювання оптимальних критеріїв розпізнавання та застосування їх до апріорно заданих ознак про об'єкти. Як правило, в реальних задачах, перш ніж формулювати критерії розпізнавання, необхідно сформувати набір найбільш інформативних ознак, які б забезпечили адекватність процедури розпізнавання реальному об'єкту.

Стосовно задачі розпізнавання рукописного тексту в роботі [3] запропонований підхід, який грунтується на заданні непохідних елементів в структурі написання рукописної букви та реалізації процедури вертикального та горизонтального сканування літери. Очевидно, для оптимального вибору найбільш інформативних елементів в структурі рукописної букви необхідно реалізувати процедуру самонавчання системи та шляхом ітераційних наближень сформувати ознаки високоточного розпізнавання рукописних літер.

Загальний опис архітектури системи розпізнавання тексту

Перейдемо до опису загальної структури системи та виділення її підсистем. Оскільки планується, що система може реалізовувати функцію самонавчання, то доцільно зробити висновок, що вона буде тісно взаємодіяти із базою даних. Також нам будуть необхідні певні засоби (що грунтуються на інструментах, які надає той чи інший

пакет прикладного програмування), що нададуть системі можливість мати доступ до бази даних.

Для полегшення проектування системи нам необхідно відокремити основні сутності та зв'язки між ними (бізнес-об'єкти та взаємодія між ними). Це дозволить уявити схему системи і чітко визначити, яка частина за що відповідає, що сприятиме зменшенню кількості проблем у майбутньому.

Також відокремлення сутностей робить код і документацію більш легкими для розуміння, що необхідно як для розробників на етапі супроводу, переробки, модифікації системи чи її підсистем, так і потенційним користувачам системи на етапі узгодження її функціональності чи її вивчення.

Останній елемент, якому необхідно приділити увагу – це підсистема виведення інформації на перегляд кінцевим користувачам. Йдеться про інтерфейс користувача, який має бути зручним, зрозумілим та інтуїтивно легким у використанні, легконастроюваним тощо.

Загальну схему системи розпізнавання рукописного тексту відображає рис. 1.

Така схема побудови програмних засобів [4] дозволяє чітко розмежувати всі рівні роботи з інформацією (зберігання, читання, підготовка та виведення), а також в довільний момент часу, не змінюючи коду програми і навіть не перекомпільовуючи її, змінювати реалізації, що підходять під стандартні інтерфейси системи. При цьому ця гнучкість притаманна кожній із частин системи. Наприклад, в певній системі рівень доступу до бази даних забезпечує роботу із форматом баз даних *MySQL*. Через деякий час актуальною стала проблема використання замість формату *MySQL* формат баз даних *MS SQL* (тому що він більше підходить для роботи з великими об'ємами даних). Для великої системи перехід на новий формат є дуже великою проблемою. Отже, наявністю проміжного рівня доступу до бази даних із декларованими інтерфейсами, які він видає назовні своїм користувачам (іншим блокам системи), ця проблема розв'язується. Нам необхідно лише буде поміняти реалізацію цих інтерфейсів, яка забезпечує роботу із форматом *MS SQL*, а решта коду залишиться без змін. Отже, проблема буде розв'язана дуже швидко шляхом локалізації її вирішення.

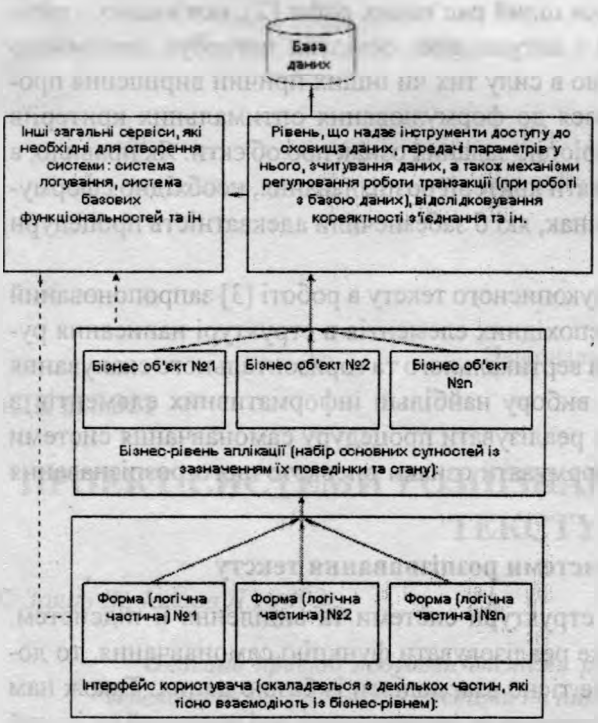

Рис. 1. Загальна схема системи розпізнавання образів

Так само можна чинити із інтерфейсом користувача. На даний момент часу користувачеві потрібний один тип інтерфейсу (наприклад, в стандарті *Windows*), а в інший

момент часу – в стандарті *Oracle*. Знову ж таки, нам добре відоме місце системи, де достатньо замінити один блок на інший для вирішення цієї проблеми. Отже, залишається тільки одна проблема – написати цей новий блок.

Крім того, різні реалізації одного і того ж інтерфейсу можуть одночасно бути присутніми у системі. За допомогою певних опцій системи вони будуть підставлятися у відповідне місце і виконувати свої функціональності, тоді як інші реалізації будуть незадіяні.

Підсистема доступу до сховища даних

Англомовна назва такого класу систем є *Persistence*, тобто щось, що є постійним. Ця система призначена для забезпечення верхніх рівнів аплікації всіма інструментами, що стосуються доступу до сховища даних. Сховищем даних можуть бути довільні формати баз даних, файли, мережеві ресурси та ін. Наприклад, для бази даних необхідно забезпечити такі функції: передача параметрів, виконання *SQL*-запитів, зчитування параметрів, керування транзакціями, забезпечення зв'язку та ін.

Підсистема доступу інкапсулює в собі загальні риси роботи зі сховищем даних і абстрагує верхні рівні від такого поняття, як сховище даних (база даних, файл). Верхні рівні нічого не знають про те, якого формату сховище даних вони використовують, що таке сховище даних, як встановити зв'язок з ним – все це знає *Persistence*-рівень.

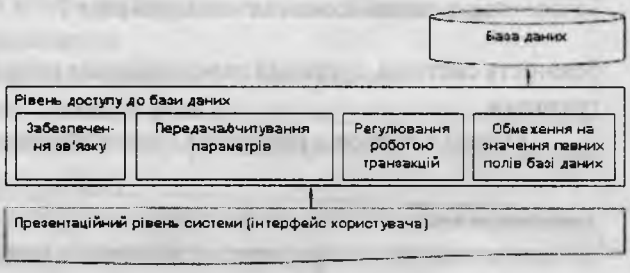

Рис. 2. Загальна схема рівня доступу до бази даних

Натомість (наприклад, при роботі із базами даних) верхній рівень знає назви сторед-процедур, які повинні ним використовуватись, тексти *SQL*-запитів, а саме виконнаня він перекладає на *Persistence*-рівень. З цього випливає, що система може в принципі використовувати різні формати баз даних не шляхом модифікації існуючого коду, а шляхом додавання все нових реалізацій класів *Persistence*-рівня.

Загальна схема рівня доступу до сховища даних наведена на рис. 2.

Бізнес-рівень системи (сукупність сутностей)

З назви цього рівня можна зрозуміти, що він використовується для дій. Ці дії стосуються саме до сутностей нашої системи. Наприклад,при роботі з базою даних можна виділити чотири основні дії (зчитування даних, додавання, вилучення, редагування) та дії, що походять від них (складні пошуки, комбіновані додавання, редагування та вилучення). При роботі із файлами чи іншими накопичувачами інформації також є актуальними ці дії з їх різновидами. Цей рівень є відображенням логіки дії всієї системи загалом, оскільки він декларує всі операції, що їх система виконує над даними. Він дуже тісно взаємодіє із рівнем доступу до даних.

Загальна схема об'єкта бізнес-рівня системи наведена на рис. 3.

Презентаційний рівень системи (інтерфейс користувача)

Презентаційний рівень може бути реалізований довільними засобами, що надає програмісту будь-яке середовище візуального проектування. Головною метою створення такого рівня є відокремлення операцій щодо підготовки даних до виведення та їх виведення від операцій обміну даними із сховищем. Основна мета його – це обробити дані, що надійшли з інших рівней і після обробки вивести їх на екран. Основна перевага такого підходу – підставляючи різні реалізації презентаційної частини аплікації, можна довільно обробляти дані і, грунтуючись на цих обробках, виводити їх на екран, причому від розробника не вимагається модифікувати вже реалізований код, а лише створювати новий. За допомогою такого підходу досягаються зручні масштабованість системи, супровід та модифікація коду, а також легке сприйняття коду користувачами .

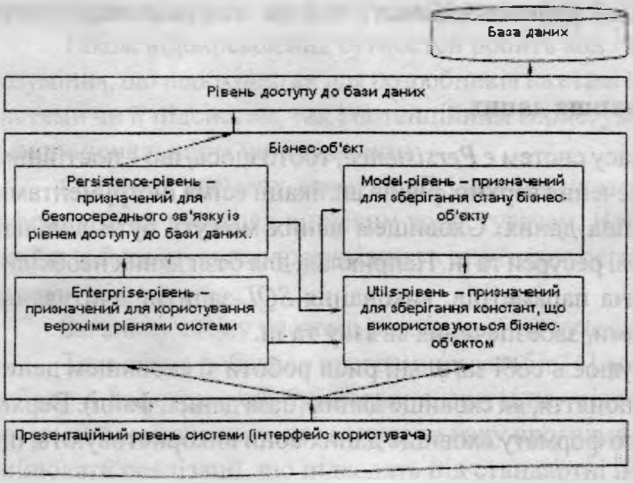

Рис. 3. Загальна схема об'єкта бізнес-рівня

Одним з підходів реалізації презентаційного рівня є *MVC*-система (модель-вид-контролер – *model-view-controller*). Сенс цієї системи в тому, що є три частини, одна з яких (*Controller*) відповідає лише за керування потоком команд, що подаються *View* (наприклад, при натисканні клавіші "Пошук" необхідно викликати відповідні методи бізнес-рівня, які забезпечують повернення даних, що були знайдені за пошуком). Ці команди *Controller* за таблицею переходів і станів переадресує на *Model*, яка і викликає бізнес-методи. Друга частина – це *Model*. Вона нічого не знає про засіб, який буде відображати дані (*View*) та про контролер. Її функція – отримувати дані від бізнес-рівня та перетворювати їх у зручний для відобра-

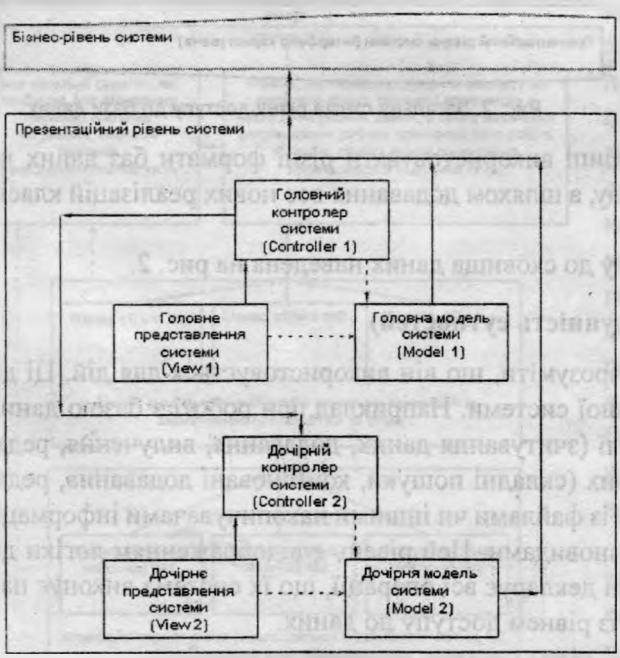

Рис. 4. Загальна схема презентаційного рівня

ження формат. Під одну і ту ж модель можна підставляти декілька *View*. Модель лише надає інтерфейс методів для *Controller* і інтерфейс своїх властивостей для *View*. Третя частина – це *View*. *View* містить лише правила виведення даних на екран. Отже, за допомогою *MVC*-моделі можна в межах презентаційного рівня відокремити процеси підготування даних до виведення і самого виведення даних кінцевому користувачеві.

Загальна схема презентаційного рівня наведена на рис. 4.

Як показано на рис.4, презентаційний рівень може складатися з декількох *MVC*-моделей: одна головна, а решта – дочірні. Це дає можливість керувати всім презентаційним рівнем як ієрархічною структурою. Наприклад, якщо виникла помилка на якомусь нижньому рівні, вона передається на найвищий рівень і, доки вона не буде виправлена, вся система не буде працювати. Такий підхід дає можливість робити надійні системи, в яких місце виникнення помилки досить легко відслідкувати.

Висновки

У статті наведено підхід до проектування системи розпізнавання рукописного тексту на основі *n*-рівневої архітектури. Кожен рівень архітектури має певні ознаки і відповідає лише за певні функціональності. Кожен рівень такої системи має свої підрівні, які грунтуються на об'єктно-орієнтованих підходах до проектування. При реалізації системи застосований системний підхід, починаючи від формулювання інформативних ознак для опису рукописних літер і закінчуючи описом об'єктно-орієнтованої технології реалізації процедури розпізнавання.

Є декілька переваг застосування такої технології проектування:

– система є легкомасштабованою, оскільки дозволяє додавати нову функціональність без внесення змін у старі, тобто нові функціональності є незалежними від старих та незалежними між собою;
– система є легкозмінюваною і не вимагає знання одним розробником всіх інших підрівнів. Тобто, один розробник відповідає за один рівень, не володіючи інформацією про інші рівні. Кожен рівень є повністю відокремлений від іншого і є самодостатнім. При менеджменті такого проекту дуже легко подолати проблему заміни розробників. Досить навчити працівника правилам побудови одного рівня без ознайомлення з іншими технологіями, і він вже здатний брати участь в її розробленні;
– модульність системи дозволяє легко змінювати її функціональність на рівні заміни модулів;
– система є легкодокументованою і зрозумілою для сторонніх людей, що не беруть участі у розробці (замовники, потенційні користувачі чи розробники, які будуть використовувати написаний код).

1. Харкевич А. А. Опознание образов // Радиотехника. – 1959. – С. 15 – 19.
2. Горелик А. Л., Скрипкин В. А. Методы распознавания образов. – М.: Высшая школа. – 1989. – 232с.
3. Алексєєв А., Заяць В., Іванов Д. Алгоритм розпізнавання символів на основі структурного підходу // Вісн. НУ "Львівська політехніка. – 2002. – № 468. – С. 129 – 133.
4. Страуструп Б. Язык программирования С++, 3-е изд./Пер. с англ. - Спб.; М.: "Невский Диалект" – "Издательство БИНОМ", 1999. – 991с.