нок функцій $t1_{qf}(n)$, $t2_{qf}(n)$, $t3_{qf}(n)$, $t4_{qf}(n)$, які зображені на рис. 1-4, вони виглядатимуть як неспадні кусково-постійні функції з приростами Δt: $\approx 6{,}5\cdot10^{-7}$*сек*, $\approx 6{,}5\cdot10^{-3}$*сек*, $\approx 7{,}4\cdot10^{-8}$*сек*, $\approx 6{,}4\cdot10^{-13}$*сек*. Загалом їхній вигляд зображено на рис. 5.

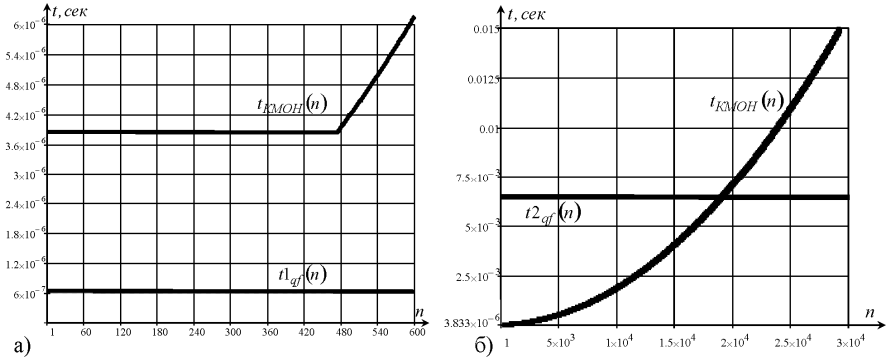

Рис. 3. Порівняння оцінок затрат часу суперкомп'ютера відповідно до КМОН типом логіки процесорних ядер з qfr -системою, квантові біти якої реалізовані на:
1 – напівпровідниковій елементній базі, 2 – НВЧ-резонаторах

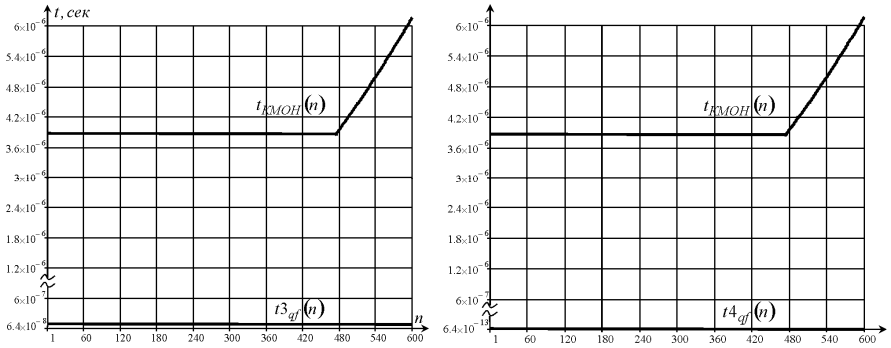

Рис. 4. Порівняння оцінок затрат часу суперкомп'ютера відповідно до КМОН типом логіки процесорних ядер з qfr -системою, квантові біти якої реалізовані на:
1 – квантових точках; 2 – іонних пастках, оптичних резонаторах або електронах-Au

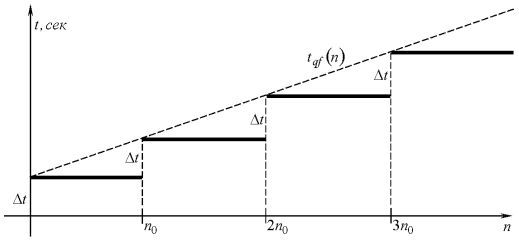

Рис. 5. Загальний вигляд оцінок витрат часу для квантового комп'ютера з різними типами елементних баз квантових біт

Висновки. З одержаних результатів випливає, що за умови малих об'ємів вхідних даних ефективними є суперкомп'ютери, а за умови великих об'ємів вхідних даних ефективними є *qfr* -системи.

Література

1. Пастух О.А. Науково-технічні основи побудови квантових нечітких обчислювальних засобів : автореф. дис. на здобуття наук. ступеня д-ра техн. наук: спец. 05.13.05 – Комп'ютерні системи та компоненти / Ін-т кібернетики ім. В.М. Глушкова НАН України. – Київ, 2013. – 38 с.

2. Пастух О.А. Чисельне моделювання процесу сумування нечітких чисел на основі квантового процесора / О.А. Пастух // Науковий вісник НЛТУ України : зб. наук.-техн. праць. – Львів : РВВ НЛТУ України. – 2009. – Вип. 19.3. – С. 260-265.

3. Пастух О.А. Числове моделювання множення нечітких чисел у квантовому процесорі / О.А. Пастух // Наукові праці Національного авіаційного університету. – Сер.: Електроніка та системи управління. – К. : Вид-во НАУ. – 2009. – № 2 (20). – С. 154-158.

Пастух О.А. **Математическое моделирование эффективности квантовых радиотехнических систем преобразования нечеткой информации**

Формально выполнено математическое моделирование эффективности квантовых радиотехнических систем превращения нечеткой информации на основе теории квантовых нечетких множественных чисел. В частности, сравнены между собой эффективность добавления и умножения нечетких числовых данных в квантовых радиотехнических системах и модели суперкомпьютера "Jaguar". Установлено, что квантовые радиотехнические системы по вычислительным критериям являются эффективнее в сравнении с самыми современными классическими суперкомпьютерными системами. Показано, что превращение нечеткой информации в квантовых радиотехнических системах является адекватнее, в сравнении с четкой информацией.

Ключевые слова: квантовая радиотехническая система, преобразование нечеткой информации.

Pastukh O.A. **The Mathematical Simulation of Quantum Radio Engineering System Efficiency at Transformation of Fuzzy Information**

The mathematical simulation of quantum radio engineering system efficiency while transforming fuzzy information is investigated. The efficiency of addition and multiplication of fuzzy digital data in quantum radio-engineering systems and "Jaguar" supercomputer model are compared in particular. Quantum radio engineering systems according to their computational criteria are proved to be more effective ones in comparison to the most modern classical supercomputer systems. The conversion of inaccurate data in quantum radio engineering systems is shown to be more adequate in comparison to accurate data.

Key words: quantum radio engineering system, transformation of fuzzy information, efficiency, data, simulation.

УДК 004.056:55(043.3)

Проф. О.М. Нечай, канд. техн. наук – Академія сухопутних військ ім. гетьмана П. Сагайдачного, м. Львів

ПРОЕКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ РОЗШИФРУВАННЯ КРИПТОГРАМ НА ОСНОВІ МЕТОДУ ІНФОРМАЦІЙНОЇ ДОШКИ

Розроблено програмне забезпечення інтелектуальної криптографічної системи, використовуючи CASE-засоби. Для побудови діаграм використано середовище моделювання Rational Software, створено інтерфейс інтелектуальної криптографічної системи з використанням CASE-засобів. Встановлено взаємодію модуля управління з окремими джерелами знань. Розроблений інтерфейс користувача, який має інтуїтивно зрозумілий дизайн, що забезпечує зручне користування програмним додатком.

Ключові слова: інтелектуальна система, криптографія, шифрування, CASE-засоби, відкритий текст, шифрований текст, шифр заміни, шифр перестановки, клас, інформаційна дошка.

Постановка проблеми. Забезпечення інформаційної безпеки є одним із найважливіших державних завдань поряд із забезпеченням обороноздатності країни, розвитком економіки, освіти та охорони здоров'я. Оскільки в сучасному світі широко використовують Інтернет для обміну інформацією, то забезпечити її захист є важливим аспектом. Він включає в себе напрями в галузях розвитку загальної теорії забезпечення інформаційної безпеки і захисту інформації різними методами, зокрема з використанням криптографічних механізмів, розробку методів і засобів захисту в системах електронного документообігу.

Огляд останніх досліджень і публікацій. Теоретичне питання та об'єктно-орієнтований аналіз, проектування з наведеними прикладами розглянуто в наукових працях **[**1-3**]**. Зокрема в **[**4-7**]** здійснено акцент на питаннях криптографії. У цих роботах автори детально розглядають концептуальні питання щодо захисту інформації, теоретичні підходи та практичні можливості застосування CASE-засобів. Однак не достатньою мірою враховано особливості розроблення і проектування програмного забезпечення інтелектуальних систем розшифрування криптограм.

Формулювання завдання дослідження. Основними завданнями нашої роботи є розробити інтелектуальну криптографічну систему, яка включає модель системи, яка створена за допомогою CASE-засобів, та програмне забезпечення інтелектуальної криптографічної системи; реалізувати можливість програми дешифрування та шифрування різними мовами.

Створити інтерфейс програми інтуїтивно зрозумілим для користувача інтелектуальної криптографічної системи за допомогою CASE-засобів. Для виконання цього завдання потрібно використати програмне забезпечення Rational Software, Microsoft Visual Studio 2008, а саме C#.

Виклад основного матеріалу

1. Проектування діаграм класів. Архітектура інформаційної дошки припускає, що на верхньому рівні абстракції знаходяться інформаційна дошка, декілька джерел знань і модуль управління. Інформаційна дошка є складною структурою з кількома рівнями абстракції. Абстракції реалізуються у вигляді об'єктів, що виникають на дошці в ієрархічному порядку. Ієрархічна структура об'єктів повторює різні рівні абстракцій, на яких розташовані джерела знань. Джерела знань використовують інформаційну дошку як глобальне джерело вхідних даних, часткових рішень, альтернатив, остаточних рішень і інформації. Для того, щоб почати розробку ієрархічної структури інформаційної дошки, виділимо такі три класи: **Sentence –** Повна криптограма, **Word –** Окреме слово в криптограмі та **CipherLetter –**Окрема буква в слові. Джерела знань повинні мати доступ до загальної інформації про всі припущення,. тому необхідно включити в ієрархію такий клас: **Assumption – Припущення**, зроблене джерелом знань. Потрібно знати припущення про літери у відкритому і зашифрованому тексті, які роблять джерела знання. Для цього вводиться наступний клас: **Alphabet –** алфавіт вихідного тексту, алфавіт криптограми та відповідність між ними.

Між цими п'ятьма класами є спільні риси: всі ці класи відповідають об'єктам інформаційної дошки і цим значно відрізняються від інших об'єктів, наприклад, від джерел знань і модуля управління. З цієї причини в ієрархію включається суперклас для всіх раніше перерахованих об'єктів. Попередню структуру абстракції Blackboard (Інформаційна дошка) показано на рис. 1.

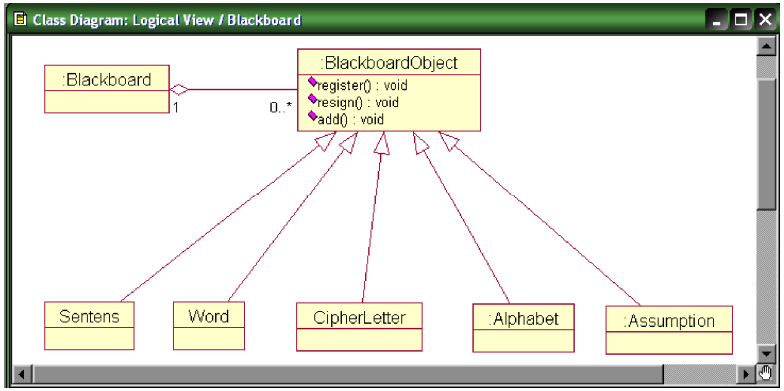

Рис. 1. Діаграма класів для абстракції B1ackboard

Окремі речення, слова і букви шифру також пов'язані між собою: у них є відповідні джерела знань. Конкретне джерело знань може проявляти інтерес до одного або кількох таких об'єктів, і тому пропозиція, слово і букви шифру повинні підтримувати зв'язок зі своїм джерелом знань, щоб при появі припущення про зміну об'єкта ці джерела знань отримували сповіщення. Для реализации этого механизма Для реалізації цього механізму вводиться простий абстрактний клас: **Dependent** (Залежний).

Класи CipherLetter (Буква шифру) і Alphabet (Алфавіт) мають ще одну спільну властивість: об'єкти цих класів допускають припущення про свою поведінку (нагадаємо, що об'єкт класу Assumption (Припущення) є одним з об'єктів класу BlackboardObject). Наприклад, деяке джерело знань може припустити, що буква К у шифрі відповідає букві Р вихідного тексту. У міру виконання завдання можна абсолютно точно з'ясувати, що буква G відповідає букві J. Отже, нам потрібен один клас, що підтримує припущення і твердження щодо пов'язаних із ним об'єктів, цей клас називається Affirmation (Затвердження).

У нашій архітектурі допускаються висловлення тільки про окремі букви – об'єкти класів CipherLetter і Alphabet. У раніше розглянутому сценарії букви шифру відповідали окремим буквам відкритого тексту, що допускають висловлювання, а алфавіт складався з багатьох букв, кожна з яких може допускати свої власні висловлювання. Визначення незалежного класу Affirmation дає змогу розділити цю властивість між цими двома класами.

Діаграма класів, представлена на рис. 2, ілюструє взаємодію між класами Dependent і Affirmation. Необхідно звернути особливу увагу на ролі, які відіграють перераховані абстракції в різних поєднаннях. Наприклад, об'єкт класу KnowledgeSource (Джерело знань) в одному аспекті може генерувати припущення (creator), а в іншому – посилатися на букву шифру (ref erencer). Оскільки

ролі змінюють зовнішній вигляд абстракцій, варто очікувати, що протокол взаємодії джерел знань і припущень буде відрізнятися від протоколу взаємодії між джерелами знань та літерами.

Маємо об'єкт класів BlackboardObject і Dependent, що представляє собою список слів. Суперклас Dependent зробимо абстрактним, оскільки можуть існувати підкласи класу Sentence, що одночасно є похідними від класу Dependent. Залишивши це відношення наслідування абстрактним, ми можемо такими підкласами розділяти властивості з суперкласом Dependent.

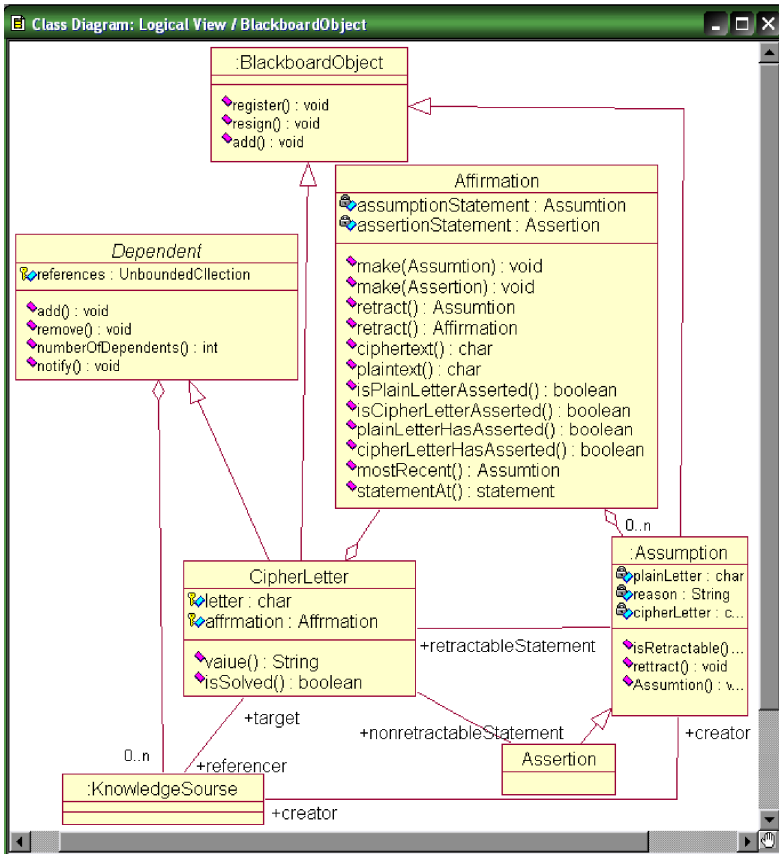

Рис. 2. Класи залежностей і підтверджень

Представники цього класу відносяться до класів BlackboardObject і Dependent. Окрім їх успадкованих властивостей, кожен об'єкт класу CipherLetter має значення, що представляє собою букву в зашифрованому тексті, а також набір припущень і тверджень про відповідні букви вихідного тексту. Для організації цього набору можна використовувати клас Affirmation.

У клас Alphabet необхідно включити захищений член affirmations і описати операції доступу до його стану. Тепер можна визначити клас Blackboard,

який зберігає колекцію примірників класу BlackboardObject і його підкласів. Таким чином, клас BlackboardObject є різновидом класу DynamicCollection (Динамічна колекція). Ми вирішили використовувати спадкування, а не включення екземпляра класу DynamicCollection, оскільки клас Blackboard задовольняє умовам використання наслідування. По суті, клас Blackboard представляє собою різновид колекції.

2. Проектування джерел знань. Можна розробити структуру абстрактного класу, що описує джерела знань.

Припустимо, що існує абстрактний клас KnowIedgeSource (Джерело знань), аналогічний класу BlackboardObject. Замість визначення всіх джерел як безпосередніх підкласів одного більш загального класу, корисно спочатку провести предметний аналіз і визначити – чи не групуються джерела знань якихнебудь чином. Дійсно, такі групи є: деякі джерела знань оперують цілими реченнями, а інші – цілими словами, безперервними ланцюжками літер або окремими літерами.

Для кожного з цих класів можна визначити спеціалізовані підкласи. Зокрема деякі підкласи класу SentenceKnowledgeSource виглядають так:

- SentenceStructureKnowledgeSource – Правила, пов'язані зі структурою пропозицій;
- SolvedKnowledgeSource – Знайдене пропозицію криптограми.
- Аналогічно, підкласи проміжного класу WordKnowledgeSource визначаються так:
- Word StructureKnowledgeSource – Правила, пов'язані зі структурою слів.

Усі ці дії не залежать від виду джерела знань. Продовжуючи узагальнення, можна зазначити, що всі вони характерні для певного механізму логічного висновку. Отже, необхідно визначити клас InferenceEngine (Генератор логічного висновку), який, маючи певний набір правил, або виконує їх, або генерує нові правила (пряма послідовність міркувань), або доводить деяку гіпотезу (зворотна послідовність міркувань). Розробляючи конструктор класу InferenceEngine, основну увагу варто приділити створенню екземпляра цього класу та ознайомлення його з набором правил, що застосовуються для оцінки припущень.

Опишемо взаємодію джерел знань. Кожне спеціалізоване джерело знань визначає свої власні правила і делегує відповідальність за їх виконання на клас InfersnceEngine. Точніше кажучи, операція KnowledgeSource: evaluate викликає операцію InferenceEngine: evaluate, що призводить до виконання однієї з чотирьох зазначених вище операцій. На рис. 3 представлено сценарій такої взаємодії.

Ця діаграма послідовностей ілюструє такі етапи сценарію:

1. Вибрати об'єкт класу KnowledgeSource.
2. Попросити його оцінити стан об'єкта класу Blackboard.
3. Виконати певну операцію, наприклад видалити об'єкт класу Assumption (скасувати припущення).
4. Повідомити всіх об'єктів класу KnowledgeSource про видалення об'єкта класу Assumption (скасування припущення).
5. Повідомити об'єкта класу Controller про те, що об'єкт класу KnowledgeSource має нову підказку, що допомагає розв'язати задачу.

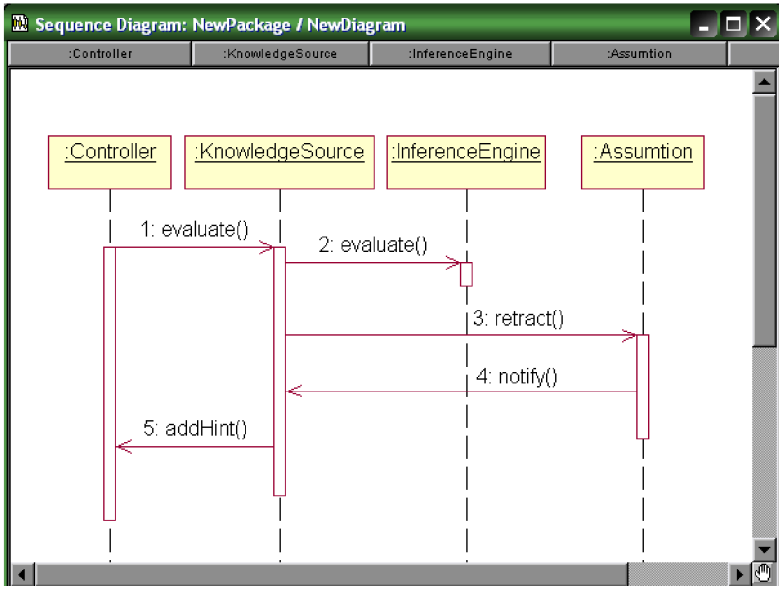

Рис. 3. Сценарій виконання правил джерелами знань

Отже, джерело знань про суфікси може виконати це правило за допомогою алгоритму зіставлення зі шаблоном і розпізнати, наприклад, шаблон * I??. Цьому шаблону можуть відповідати суфікси ING, IES І IED. Щодо структури класу, то джерела знань є різновидом механізму логічного висновку. Крім цього, вони пов'язані з об'єктами дошки, оскільки діють на об'єкти, розташовані на ній. Тоді кожне джерело знань буде пов'язане з модулем управління, якому він посилає свої підказки, а той, водночас, може час від часу активізувати джерела знань. Екземпляри класу Blackboard є сховищем об'єктів класу BlackboardObject. З аналогічних причин необхідний також клас KnowledgeSources, що зберігає всі джерела знань, пов'язані з розв'язуваною задачею.

Розглянемо взаємодію модуля управління з окремими джерелами знань. На кожному етапі розшифровки криптограми окремі джерела знань можуть з'ясувати корисну інформацію та повідомити підказку контролеру. І навпаки, джерело знань може дійти висновку, що попередня підказка була неправильною і її треба скасувати. Оскільки всі джерела знань мають рівні права, модуль управління повинен вибрати найбільш перспективну підказку і викликати операцію evaluate.

Для активізації модуля управління необхідно, щоб:

- Об'єкт класу Assertion має вищий пріоритет, ніж об'єкт класу Assumption.
- Об'єкт класу SolvedKnowledgeSource дає найбільш цінні підказки.
- Підказки об'єктів класу PatternMatchingKnowledgeSource мають більш високий пріоритет, ніж підказки об'єктів класу Sentence-StructureKnowledgeSource.

Модуль управління діє як агент, відповідальний за взаємодію між різними джерелами знань, пов'язаними з інформаційною дошкою. Модуль управління повинен мати зв'язок з джерелами знань, що забезпечується класом KnowledgeSources. Крім цього, одну зі своїх властивостей він повинен мати колекцію підказок, упорядкованих за пріоритетом. Цим самим модуль управління може легко вибрати для активізації джерело знань з найбільш цікавою підказкою.

Для опису його динамічної поведінки добре підходять кінцеві автомати.

Фізичне утримання об'єктів дошки в колекції theBlackboard та джерел знань в колекції theKnowledgeSources показано схематично, аналогічно тому, як це було зроблено при описі вкладеності класів. Клас Cryptographer агрегує дошку, джерела знань і модуль управління. Додаток може створювати кілька екземплярів цього класу і працювати з декількома інформаційними дошками одночасно. У класі Criptography визначено дві основні операції:

- reset – Перезапустити інформаційну дошку;
- decipher – Розшифрувати задану криптограму.

Відповідний потік управління зображено на рис. 4.

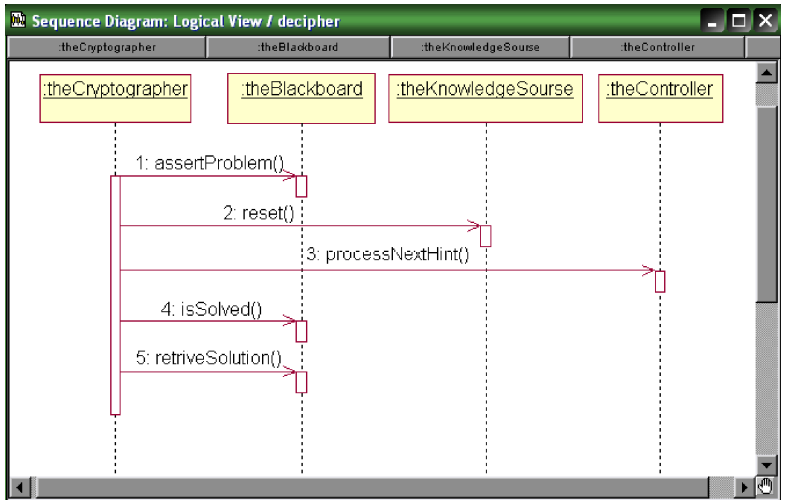

Рис. 4. Діаграма послідовностей decipher

Завершимо опис системи архітектурними інтерфейсами, необхідними для виконання алгоритму розшифровки криптограми. Розглянемо дві основні операції, визначені у класі decipher, а саме: assertProblem і retrieveSolution (Знайти рішення). Операція assertProblem особливо цікава тим, що створює всю сукупність об'єктів класу Blackboard.

3. Інтерфейс програмного додатку. Розроблений інтерфейс (рис. 5,6) користувача має інтуїтивно зрозумілий дизайн, що забезпечує зручне користування програмним додатком.

Висновки. Отже, спроектовано інтелектуальну систему розшифрування криптограм на основі методу інформаційної дошки, що імітує людський спосіб розв'язання задачі, розроблено програмне забезпечення криптографічної системи. Побудовані UML діаграми з використанням CASE-засобів.

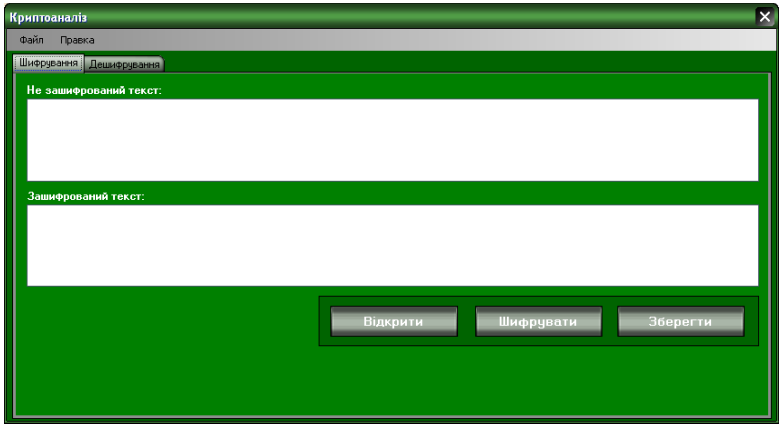

Рис. 5. Інтерфейс програмного засобу для шифрування та дешифрування тексту

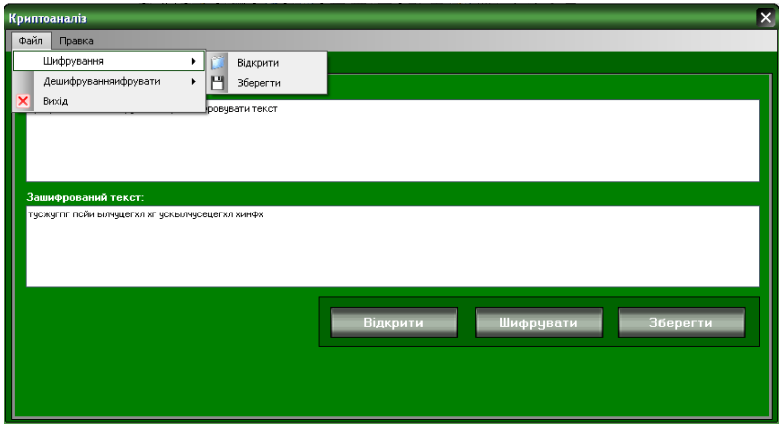

Рис. 6. Інтерфейс програмного засобу, що демонструє можливість завантаження та збереження шифрованого та дешифрованого тексту

Література

1. Гради Буч. Объектно-ориентированный анализ и проектирование с примерами приложений Object-Oriented Analysis and Design with Application / Гради Буч, Роберт А. Максимчук, Майкл У. Энгл, Бобби Дж. Янг, Джим Коналлен, Келли А. Хьюстон. – СПб. : Изд-во "Вильямс", 2008. – 720 с.

2. Петцольд Ч. Программирование с использованием Microsoft Windows Forms. Мастер-класс : пер. с англ. / Ч. Петцольд. – М. : Изд-во Русская Редакция; СПБ. : Изд-во "Питер", 2006. – 432 с..

3. Петцольд Ч. ПЗЗ Программирование для Microsoft Windows на C# : пер. с англ. / Ч. Петцольд. – В 2-х томах. – М. : Изд.-торг. дом "Русская Редакция". – 2002. – Т. 1. – 576 с.

4. Шнайер Б. Прикладная криптография. Протоколи, алгоритми, тексти програм мовою С. / Брюс Шнайер. – М. : Изд-во ТРІУМФ, 2003. – 816 с.

5. Вербіцький О.В. Вступ до криптології / О.В. Вербіцький. – Львів : Вид-во наук.-техн. літ-ри, 1998. – 236 с.

6. Salomaa A. Public – Key Cryptography / A. Salomaa. – New York : Springer-Verlag, 1996. – 236 p.

7. Jaworski J. Java Security Handbook / J. Jaworski and P. Perrone // SAMS Publishing, 2000. – 482 p.

8. Nyberg K. Differentially uniform mappings for cryptography / K. Nyberg // Lect. Notes Comput. Sci. – 2006. – Vol. 765. – Pp. 55-64.

Нечай О.М. **Проектирование интеллектуальной системы расшифровывания криптограмм на основе метода информационной доски**

Проведен обзор современного криптоанализа, разработано программное обеспечение интеллектуальной криптографической системы, используя CASE-средства. Для построения диаграмм использована среда моделирования Rational Software, создана модель интеллектуальной криптографической системы с использованием CASE-средств и проведенный анализ программного обеспечения для шифровки и дешифрации тексту с помощью шифра замены и перестановочного шифра. Установлено взаимодействие модуля управления с отдельными источниками знаний. Разработанный интерфейс, который имеет интуитивно понятный дизайн, обеспечивает удобное пользование программным приложением.

Ключевые слова: интеллектуальная система, криптография, шифровка, CASE-средства, открытый текст, шифрованный текст, шифр замены, шифр перестановки, класс, информационная доска.

Nechay O.M. **The Design of the Intellectual System of Cryptogram Decoding on the Basis of the Informative Board Method**

The review of modern crypto analysis is conducted. The intellectual cryptographic system software is developed using CASE-tools. The environment of Rational Software design is used in order to construct diagrams. The model of the intellectual cryptographic system is created with the use of CASE-tools and the analysis of software is conducted for text enciphering and decoding by the substitution code and the rearrangement code. The interaction of the management module with separate sources of knowledge is stated. A designed interface that has intuitive conceptual design provides comfortable use of software.

Key words: intellectual system, cryptography, encryption, CASE-tools, plain text, encrypted text, substitution code, rearrangement code, class, informative board.

УДК 658.1:657.1

Ст. викл. М.В. Плекан, канд. екон. наук;
аспір. І.І. Жигало – НУ "Львівська політехніка"

ІСТОТНІСТЬ ІНФОРМАЦІЙНОЇ БАЗИ У ЗАБЕЗПЕЧЕННІ ЕФЕКТИВНОСТІ ЕКОНОМІЧНОГО ІНСТРУМЕНТАРІЮ

Наведено значення та істотність вхідних джерел інформації для об'єктивності результатів і висновків економічного інструментарію для цілей обґрунтування управлінських рішень. Розглянуто існуючу проблематику під час формування задовільної інформаційної бази для вироблення адаптивності оцінно-аналітичних систем і моделей, їх дієвості у практичному впровадженні на підприємстві. Обґрунтовано, що існуюча система агрегування наявної інформаційної бази спричиняє невідповідність концептуально методологічної основи і методик економічного інструментарію, втрати ним дієвості в забезпеченні ефективності системи управління. Запропоновано окремі підходи до підвищення рівня формалізації вхідної інформаційної бази економічного інструментарію.

Ключові слова: економічний інструментарій, інформаційне забезпечення, база даних, джерела інформації, оцінка, показники, результати.

Вступ. Зростання рівня конкуренції на глобальних ринках технологій, товарів, капіталу та інвестицій спричинюють зростання комерційної таємниці та конфіденційності значної кількості інформаційних потоків, посилюючи проблематику формування необхідного обсягу та релевантності інформаційної