УДК 519.683

А. О. Шемет, аспірант Академії митної служби України

ДЕЯКІ АСПЕКТИ АВТОМАТИЗОВАНОГО ГЕНЕРУВАННЯ АЛГОРИТМІВ І ПРОГРАМ У СФЕРІ РЕЗЕРВУВАННЯ ФАЙЛОВИХ СИСТЕМ

У статті розглядаються деякі аспекти застосування автоматизованого програмування та генерації алгоритмів. Формулюється модель автоматизованого програмування, пропонується поєднання апарату, який грунтується на семантичних мережах та системі алгоритмічних алгебр.

In the article are observed some aspects of application of the automated programming and generating of algorithms. There was formulated model of automated programming system and was offered the union of the apparatus grounded on semantic networks and systems of algorithmic algebra.

Ключові слова. Автоматизація програмування, генерація алгоритмів, семантичні мережі.

Вступ. Принципи розробки програмного забезпечення, які застосовуються в цей час (включаючи поширені об'єктно-орієнтовані методи аналізу й проектування), спрямовані на розробку конкретної системи, з розрахунком на певну задачу й фіксований контекст. Навпаки, інженерія предметної області орієнтується на створення програмних продуктів багаторазового застосування.

Однак у цей момент широко вживають спроби автоматизувати процес створення алгоритмів і власне кодування (запис алгоритмів кінцевою мовою програмування). Одним з методів автоматизації алгоритмів та програм є генеративне (породжуюче) програмування.

Генеративне програмування, – парадигма розробки програмного забезпечення, в основі якої лежить ідея трансформації моделей програмного забезпечення в напрямку від моделей високого рівня, що описують предметну область програмного додатка (наприклад, властивості й відносини об'єктів автоматизації), до моделей реалізації (наприклад, кінцевий код додатка) [1].

У рамках генеративного програмування існує ряд моделей систем для автоматичного генерування алгоритмів і програмного коду, серед яких системи, котрі грунтуються на семантичних мережах.

Постановка завдання. Основна мета даної статті – формулювання вимог до моделі генеративного програмування із застосуванням системи алгоритмічних алгебр, застосування цієї моделі у сфері резервування інформації.

Застосування апарату генеративного програмування при створенні систем резервування інформації розширює можливості для створення формального апарату методів резервного копіювання.

Система автоматизованої генерації алгоритмів (САГА) для систем резервного копіювання являтиме собою програмний комплекс, що базується на знаннях. Виконання генерації відбуватиметься за таким планом:

– аналіз тексту задачі на обмеженій природній мові (формування вимог до програми резервного копіювання);

– побудова плану розв'язання задачі (розбір виразів природної мови й зіставлення їх з базою знань);

– визначення оптимального методу резервного копіювання для поставленої задачі;

© А. О. Шемет, 2009

– складання алгоритмів за цим планом згідно з обраним методом (формування коду алгоритму);

– аналіз трудомісткості генерованих алгоритмів (відбір менш трудомісткого алгоритму);

– перевірка алгоритму на можливість розв'язання.

Як базу знань пропонується використати бінарну неоднорідну семантичну мережу. Семантична мережа – інформаційна модель предметної області, що має вид орієнтованого графа, вершини якого відповідають об'єктам предметної області, а дуги (ребра) задають відносини між ними [2]. Об'єктами можуть бути поняття, події, властивості, процеси. Під об'єктами предметної області розумітимемо факти, що ставляться до області резервного копіювання ("джерело", "приймач", "файл", "директорія", "диск" і т. д.). Під відносинами будемо розуміти такі: "належить", "використовує", "доповнює" й т. д.

На рис. 1 зображено спрощену схему стадій розробки додатків [3]. Модель розробки програмного забезпечення (каскадна, спіральна) в даній статті не розглядається, а лише наводяться основні етапи. Розробка додатку починається з формування вимог до системи (програмного забезпечення,) які разом з розробником формуються у технічне завдання. Наступний етап – розробка проекту (діаграми, моделі тощо). Після створення пакету проектної документації всі діаграми переводяться на конкретну мову програмування (кодування). Закодований проект проходить етапи компіляції, зборки, тестування і впровадження [4].

Збирання
і компіляція
програми

Упровадження

Налагодження
і тестування

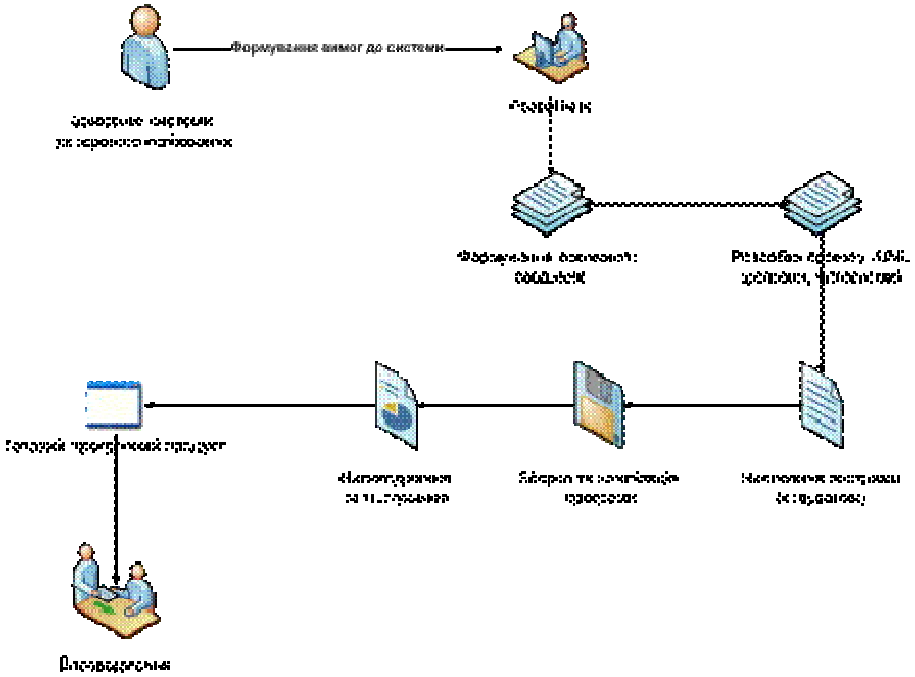

Рис. 1. Етапи розробки додатків

У запропонованій системі автоматизованого програмування пропонується автоматизувати процес створення проектної документації та етап написання коду програми (рис. 2). Тобто планується робити перехід від слабоформалізованих висловлювань до формалізованих алгоритмів, а у свою чергу, до мови програмування.

Збирання
і компіляція
програми

Упровадження

Налагодження
і тестування

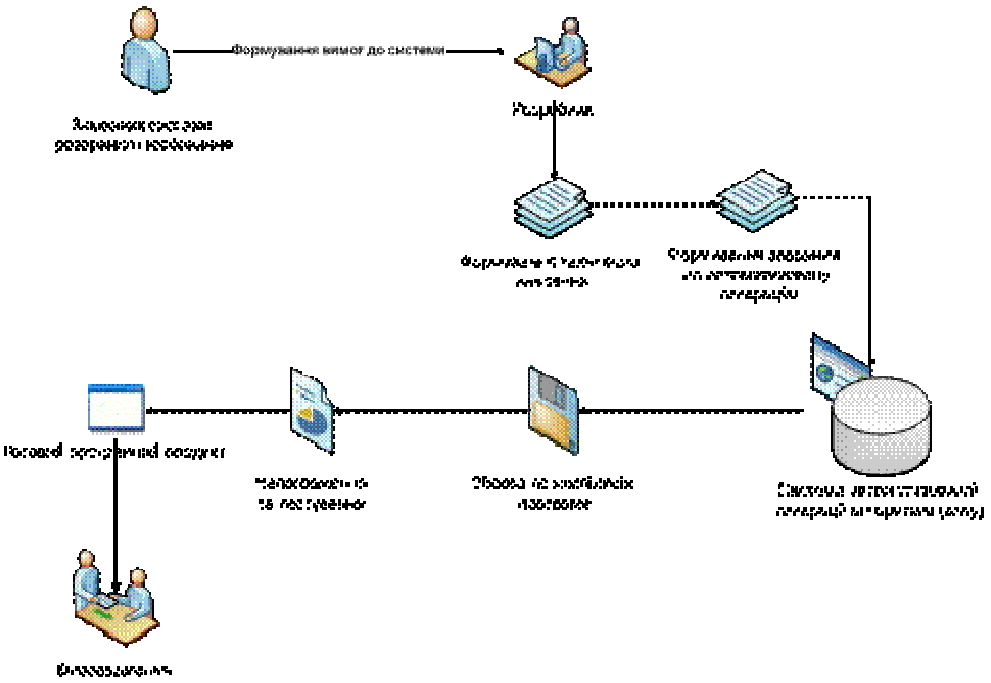

Рис. 2. Розробка додатків за допомогою системи автоматизованої генерації алгоритмів для систем резервування інформації

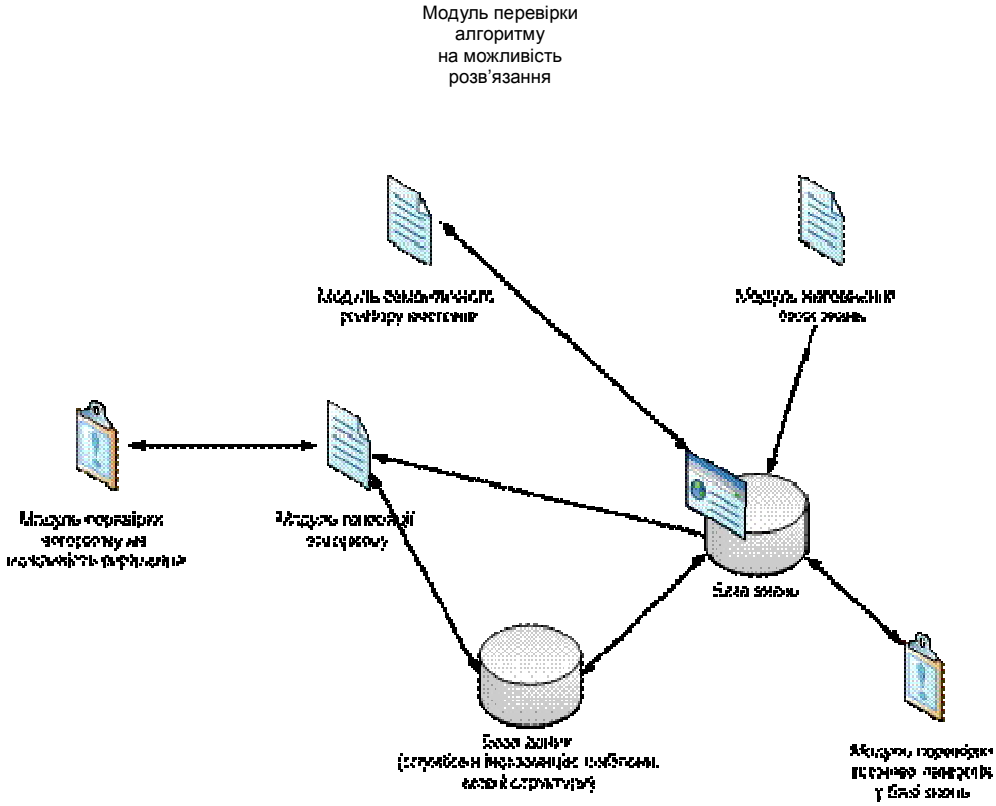

Рис. 3. Модель системи автоматизованого програмування

На рис. 3 наведено спрощену модель системи автоматизованого програмування. В основі запропонованої моделі лежить семантична мережа, але її застосування планується використовувати тільки для переходу від неформалізованої мови (висловів типу *"достигнут конец списка файловой системы образа"*) до слабоформалізованого алгоритму, який також записується обмеженою мовою.

Для переходу від слабоформалізованих структур до формалізованої мови програмування будемо застосовувати методи породжуючого програмування (Generative Programming) (*А. Е. Дорошенко, Л. М. Захарія, Г. Е. Цейтлін)* [5] та систем алгоритмічних алгебр (САА), запропонованих В. М. Глушковим [6].

Результати дослідження. Як приклад розглянемо цей перехід на алгоритмі для методу повного резервного копіювання. Для цього спочатку запишемо алгоритм звичайною мовою.

НАЧАЛО ПРОЦЕДУРЫ
ЕСЛИ “количество образов равно 0”
ТО “выйти из процедуры”
ИНАЧЕ “перейти к циклу”
ПОКА НЕ “достигнут конец образа(А)”
ЦИКЛ
“Прибавить счетчик образов на 1”
ПОКА НЕ “достигнут конец списка файловой системы образа”
ЦИКЛ
СРАВНИТЬ “файловую систему источника” И “файловую систему образа (А)”
ЕСЛИ “объект в источнике равен удалено”
ТО “Копировать объект в буфер (удалено)”
“Увеличить буфер на 1”
ИНАЧЕ
ЕСЛИ “объект в источнике не равен объекту в образе(А)”
ТО “копировать объект в буфер (модифицировано)”
“Увеличить буфер на 1”
ИНАЧЕ
ЕСЛИ “объект не присутствует в образе(А)”
ТО “копировать объект в буфер (добавлено)”
“Увеличить буфер на 1”
ИНАЧЕ
“Увеличить А на единицу”
КОНЕЦ ЦИКЛА
КОНЕЦ ЦИКЛА

ЕСЛИ “буфер (модифицировано)” ИЛИ “буфер (добавлено)” НЕ “пуст”
ТО
“Создать N-образ”
ПОКА “буфер (модифицировано)” НЕ “конец”
ЦИКЛ
“копировать объект в N-образ”
“увеличить индекс буфер (модифицировано) на единицу”
КОНЕЦ ЦИКЛА
ПОКА “буфер (добавлено)” НЕ “конец”
ЦИКЛ
“копировать объект в N-образ”
“увеличить индекс буфер (добавлено) на единицу”
КОНЕЦ ЦИКЛА
ИНАЧЕ
“обнулить счетчики”
ВЫХОД ИЗ ПРОЦЕДУРЫ

Вищенаведений запис алгоритму є слабоформалізованим. Наступний етап – присвоїмо виразам, які було використано в алгоритмі, ідентифікатори:

N001: “количество образов равно 0”
N002: “выйти из процедуры”
N003: “перейти к циклу”
V001: “достигнут конец образа(А)”
V002: “прибавить счетчик образов на 1”
V011: “достигнут конец списка файловой системы образа”
V012: СРАВНИТЬ “файловую систему источника” И “файловую систему образа (А)”
N011: “объект в источнике равен удалено”
N012: “копировать объект в буфер (удалено)”
N013: “увеличить буфер на 1”
N021: “объект в источнике не равен объекту в образе(А)”
N022: “копировать объект в буфер (модифицировано)”

N023: “увеличить буфер на 1”
N031: “объект не присутствует в образе(А)”
N032: “копировать объект в буфер (добавлено)”
N033: “увеличить буфер на 1”
N034: “увеличить А на единицу”
N041: “буфер (модифицировано)” ИЛИ “буфер (добавлено)” НЕ “пуст”
N042: “создать N-образ”
V021: “буфер (модифицировано)” НЕ “конец”
V022: “копировать объект в N-образ”
V023: “увеличить индекс буфер (модифицировано) на единицу”
V031: “буфер (добавлено)” НЕ “конец”
V032: “копировать объект в N-образ”
V033: “увеличить индекс буфер (добавлено) на единицу”
N043: “обнулить счетчики”

Де *N*xxx – розгалуження, *V*xxx – прохід по циклу
Запишемо алгоритм за допомогою ідентифікаторів:

```
НАЧАЛО ПРОЦЕДУРЫ
ЕСЛИ N001
        ТО N002
        ИНАЧЕ N003
        ПОКА НЕ V001
            ЦИКЛ
                V002
                ПОКА НЕ V011
                ЦИКЛ
                СРАВНИТЬ V012
                ЕСЛИ N011
                ТО N012
                N013
                ИНАЧЕ
                ЕСЛИ N021
                ТО N022
                    N023
                ИНАЧЕ
                ЕСЛИ N031
                ТО N032
                N033
                ИНАЧЕ
                N034
                КОНЕЦ ЦИКЛА
        КОНЕЦ ЦИКЛА
        ЕСЛИ N041
        ТО
            N042
              ПОКА V021
              ЦИКЛ
            V022
            V023
              КОНЕЦ ЦИКЛА
              ПОКА V031
              ЦИКЛ
            V032
            V033
              КОНЕЦ ЦИКЛА

        ИНАЧЕ
```

N043
ВЫХОД ИЗ ПРОЦЕДУРЫ

Проводячи покрокову деталізацію вищенаведених конструкцій, можемо дійти найпростіших примітивів, звернувшись до бази даних для зіставлення конструкцій з реальними сегментами мови програмування.

Висновок. Отже, поєднання систем, які грунтуються на семантичних мережах з апаратом системи алгоритмічних алгебр, дозволяє створити ефективний механізм переходу від неформалізованих висловів до чітко формалізованої мови програмування.

Застосування вищенаведеної моделі автоматизованого створення програм у сфері резервування інформації забезпечить чітку формалізацію методів роботи з файловими потоками.

Література

1. Чарнецки К. Порождающее программирование: методы, инструменты, применение. Для профессионалов [Текст] / К. Чарнецки, У. Айзенекер. – СПб. : Питер, 2005. – 736 с.

2. Корнеев В. В. Базы данных. Интеллектуальная обработка информации [Текст] / В. В. Корнеев, А. Ф. Гареев, С. В. Васютин, В. В. Райх. – М. : Нолидж, 2000.

3. Братищенко В. В. Проектирование информационных систем [Текст] / В. В. Братищенко. – Иркутск : БГУЭП, 2004. – 84 с.

4. Стандарт ISO/IEC 12207:1995 “Information Technology – Software Life Cycle Processes”.

5. Ющенко Е. Л. Многоуровневое структурное проектирование программ. Теоретические основы, инструментарий [Текст] / Е. Л. Ющенко, Г. Е. Цейтлин, В. П. Грицай, Т. К. Терзян. – М. : Финансы и статистика, 1989.

6. Глушков В. М. Алгебра. Языки. Программирование [Текст] / В. М. Глушков, Г. Е. Цейтлин, Е. Л. Ющенко. – К. : Наук. думка, 1974. – 328 с.