**МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ АВІАЦІОННИЙ УНІВЕРСИТЕТ**

***«Реінженерія програмного забезпечення авіаційних тренажерів»***

**СИДОРОВ Євген Миколайович** - кандидат технічних наук, доцент кафедри комп’ютерних інформаційних технологій.

**реферат**

**Київ – 2011**

Розглянуто зворотну інженерію програмного забезпечення, як важливу складову реінженерії, зроблено аналіз існуючих методів та засобів зворотної інженерії, визначено взаємозв’язок між зворотною та прямою інженеріями в контексті реінженерії, наведено характеристику технічно-програмного комплексу авіаційного тренажеру, на прикладі якого виконувалась перевірка працездатності запропонованого методу.

Виконання процесів супроводження програмного забезпечення призвело до необхідності реконструювання програм та розробки відповідного розділу інженерії програмного забезпечення, який називається зворотною (backward) або реверсивною (reverse) інженерією. Підхід, який забезпечує процеси розробки програмного забезпечення називається прямою (forward) інженерією. Задача зворотної інженерії протилежна задачі прямої інженерії та полягає в забезпеченні процесів отримання з низькорівневого представлення програмного забезпечення (похідного коду), його високорівневого представлення, наприклад, проектної інформації, або специфікацій вимог. Тому, основна мета зворотної інженерії – відновлення інформації про програмне забезпечення, а підставою для формулювання цієї мети є необхідність розуміння успадкованого програмного забезпечення. Так як, програмне забезпечення майже завжди представлено похідним кодом, то усі відомі методи та засоби відновлення інформації про програмне забезпечення орієнтовані саме на похідний код.

Реінженерія (reengineering) програмного забезпечення – це підхід, що забезпечує переробку програмного забезпечення, ґрунтуючись на досвіді відповідного домену, та будується шляхом взаємозв’язку зворотної та прямої інженерій («кругообіг» програмного забезпечення) (рис. 1).



Рис. 1. «Кругообіг» програмного забезпечення

Сьогодні відомі роботи, в яких пропонуються методи вирішення задачі реінженерії, але в окремих доменах (D.Garlan), або для деяких типів програмного забезпечення (S.Rugaber, T.Mens). Засоби зворотної інженерії, як правило, будуються на основі відомої схеми (E.Chikofsky, J.Сross). Застосування загальних методів, наприклад, відомого методу відновлення проектної інформації (design recovery, T.Biggerstaff) потребують врахування особливостей виконання процесів реінженерії, які залежать від домену. Тому, досвід показує, що кожен новий домен, як правило, вимагає відповідних досліджень та доробки існуючих методів реінженерії, або розробки нового методу.

На прикладі програмно-технічного комплексу авіаційного тренажеру, наведено характеристики домену в якому було застосовано розроблений метод. Запропонований метод керованої об’єктом реінженерії програмного забезпечення.

Програмне забезпечення, яке досліджується та підлягає реінженерії, є частиною реального світу, функціонує на цифрових обчислювальних комплексах, та значна кількість його компонентів пов'язана з обробкою інформації, циркулюючої в імітаційній моделі. До складу таких комплексів входять аналого-цифрові та цифро-аналогові перетворювачі, датчики реального об'єкту або його моделі, інше обладнання. Особливості домену та місця тренажеру в ньому визначають те, що реінженерія успадкованого програмного забезпечення цього типу окрім вирішення традиційних завдань, пов'язаних з відновленням проектної інформації, вимагає відновлення інформації про реальний об'єкт та особливого підходу до рішення задачі обґрунтування адекватності функціонування побудованого в результаті реінженерії програмного забезпечення поведінці реального об'єкту. Інформація про реальний об'єкт, відновлювана в процесі реверсивної інженерії, – це алгоритми, моделі, характеристики вхідних параметрів, датчиків, приладів та виконавчих пристроїв або моделі реального об'єкту. При відновлюванні, окрім шляху для отримання інформації – аналізу успадкованого коду та документації – слід використовувати експериментальне дослідження поведінки моделі.

Обґрунтування адекватності функціонування побудованого програмного забезпечення не може здійснюватися відомими методами – порівнянням результатів виконання успадкованого та нового програмного забезпечення або порівнянням результатів реверсивної інженерії з поведінкою відповідної моделі домену. Перший метод не можна використовувати за відсутністю, внаслідок заміни обчислювального обладнання, на якому можна виконати успадковане програмне забезпечення, а другий, – оскільки модель домену, представлена в документації, як правило, містить помилки. Окрім цього, особливе положення програмного забезпечення в домені указує на те, що основна увага при обґрунтуванні адекватності повинна приділятися детальному аналізу його функціонування в реальних умовах. Все це свідчить про те, що вирішальну роль в обґрунтуванні адекватності функціонування розробленого в результаті реінженерії програмного забезпечення, функціонуванню реального об'єкта повинні грати характеристики та властивості реального об'єкту. Це складає сутність запропонованого методу керованої об’єктом реінженерії програмного забезпечення (рис. 2.).



Рис. 2. Схема методу реінженерії

Успадковане програмне забезпечення (рис. 3.) характеризує дискретний процес моделювання, а необхідна швидкодія досягалася за рахунок наступного: спрощенням математичних моделей динаміки польоту та систем літака; заміною аналітичного опису реальних (динамічних) процесів залежностями у вигляді таблиць рішень; використанням простих, але швидких методів інтерполяції функцій; використанням простих методів інтегрування диференційних рівнянь.

На рис. 4 показана загальна схема виконаної міграції програмно-технічного забезпечення тренажера. Оскільки, побудоване програмне забезпечення не можна було перевірити на правильність функціонування шляхом виконання успадкованого програмного забезпечення (обчислювач було замінено), то реверсивна інженерія, окрім традиційних процесів, включала процес додаткової взаємної перевірки успадкованого коду та моделі.



Рис. 3. Склад успадкованого програмного забезпечення

Більш всього помилок було в коефіцієнтах рівнянь моделі, що описує динаміку польоту. Тому для їх уточнення використовувався успадкований код, в якому визначалися частини, що здійснюють обчислення коефіцієнтів. Інтерпретація цих частин «за столом» дозволила визначити значення помилкових коефіцієнтів.



Рис. 4. Міграція технічного та програмного забезпечення тренажера

Реінженерія успадкованого програмного забезпечення після перевірки моделей здійснювалася в два етапи: реверсивна інженерія – побудова моделі та алгоритмічного уявлення; пряма інженерія – по отриманих моделях та алгоритмах, будувався код нового програмного забезпечення в мові С. Для виконання реверсивної інженерії за відомою схемою був побудований спеціальний інструмент, який містить екстрактор та абстрактор. Робота інструмента та результат прямій інженерії на фрагменті коду показаний на рис. 5 (розглядується модель параметра – ознаки перебування двигуна в режимі розкручування ротора стартером).

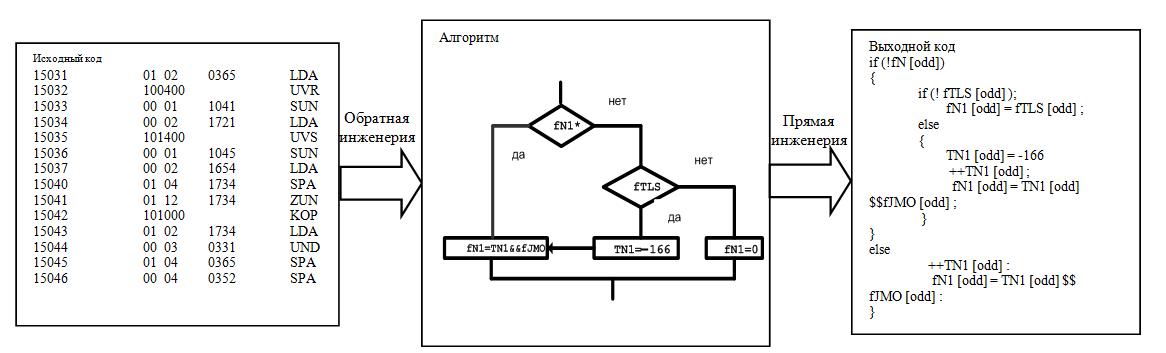


Рис. 5. Результат реінженерії (фрагмент)

Дослідження процесів реінженерії програмного забезпечення, вперше дозволило виявити декілька схем їх виконання за наступним сценарієм: *S : L  N,* де *S* – позначення схеми реінженерії;  *L –* успадковане програмне забезпечення*; N –* нове програмне забезпечення; *С –* множина умов, від яких залежить застосування схеми*.* Було встановлено чотири наступні базові схеми реінженерії програмного забезпечення: «код-код», «код-алгоритм-код», «код-модель-алгоритм-код», «код-модель-технічний опис-алгоритм-код». Схему «код-модель-технічний опис-алгоритм-код» наведено на рис.6, в якості прикладу. Умовою (С) для застосування цієї схеми є наступне: успадкований код (L) реалізує невідому модель, створену розробником, з якої зрозуміти алгоритми неможливо; опис змісту фізичного процесу, що моделюється, є в технічній документації реального об’єкту. Сполучення моделі та технічного опису дозволяє побудувати алгоритм та створити нове програмне забезпечення (*N*).



Рис. 6. Схема реінженерії

По цій схемі була проведена реінженерія більшості програм для моделювання параметрів імітаторів «силова установка» та «динаміка польоту», а також декількох параметрів імітатору «навігаційна система». Наприклад, моделюється логічний параметр  імітатору «навігаційна система» (параметр забезпечує сигналізацію при прольоті одного з маршрутних маяків). Фрагмент коду, в якому моделюється параметр  показано на рис. 7.

23440 00020646 MARK LDA HI

23441 040577 KAR 1

23442 00041650 SPA HMX

23443 00020646 LDA HI

23444 00041651 SPA HMZ

23445 00020332 LDA LG13

23446 00030115 UND MASK+S

23447 101040 UAS

23450 00011457 SUN AD1

23451 00021650 LDA HMX

23452 040577 KAR 1

23453 00041650 SPA HMX

23454 00021651 LDA HMZ

23455 040577 KAR 1

23456 00041651 SPA HMZ

23525 00120100 AD2 ZUN AA1

23526 00011530 SUN \*+2

23527 00011532 SUN AD4

23530 00120000 ZUN 0

23531 00011463 SUN AD1+4

23532 0002 0337 AD4 LDA LG23

23533 041675 KZL 3

23534 140100 VZP

23535 040675 KZP 3

23536 00040337 SPA LG23

Рис.7. Фрагмент коду, що моделює.

Математична модель, отримана в результаті аналізу приведеного вище коду має вигляд: , де ХТ – подовжна координата літака; ZT – бокова координата літака; XMJ – подовжна координата маркера; ZMJ – бокова координата маркера; HMX – дорівнює половині висоти польоту; HMZ – дорівнює висоті польоту. З урахуванням отриманої математичної моделі параметра та успадкованого коду був побудований алгоритм (рис.8).



Рис. 8. Алгоритм моделювання параметру.

В мові С алгоритм показано на рис. 9.

HMX = 0.5 \* HI;

HMZ = HI;

if (fCM == 1)

{

HMX = HMX \* 0.5;

HMZ = HMZ \* 0.5;

}

for(marker = 2; marker < 5; ++marker)

{

XT = 16 \*(XT – XMJm);

ZT = 16 \* (ZT –ZMJm);

if (XT < HMX)

if (ZT < HMZ)

{

fMRT = 1;

break;

}

else

fMRT = 0;

else

fMRT=0;

}

Рис. 9. Код алгоритму моделювання параметру в мові С.

Для аналізу похідного коду (мова програмування SYPS платформи Robotron), та побудови його високорівневого алгоритмічного уявлення на платформі Microsoft.Net версії 3.5 та мові програмування C# було побудовано інструмент зворотної інженерії (за архітектурою «екстрактор – абстрактор», E.Сhicofsky). Він має графічний інтерфейс користувача, який є Web-застосуванням. Послідовність функцій, яку виконує інструмент наступна: завантаження похідного коду для виконання аналізу; читання коду; перевірка помилок; аналіз похідного коду (екстрактор); трансляція в мову DOT (екстрактор); запис в .dot файл; завантаження файлу; виклик графічного модуля (абстрактора).

Розглянуто архітектуру засобів, які розроблялись для виготовлення тієї частини програмного забезпечення, що створюється заново, внаслідок повної заміни обчислювального комплексу,програмне забезпечення обмину, імітатору шуму та пульту інструктора. Для реалізації програмного забезпечення обмину запропоновано архітектуру програмного шаблону (рис. 10). Вона включає два рівні драйверів, які забезпечують функції передачі/прийому параметрів, конфігуратор, що забезпечує конфігурування пристроїв вводу/виводу, вхідних та вихідних параметрів, і програмне забезпечення обміну даними, що реалізує взаємодію з іншими обчислювальними вузлами тренажера.



Рис. 10. Архітектура шаблону програмного забезпечення

В архітектурі передбачається набір інструментів, які дозволяють отримувати інформацію про об'єкт в процесі зворотної інженерії, яке необхідне для перевірки правильності моделі. Сховище конфігурацій має ієрархічну структуру на основі XML, яку можна редагувати, переглядати та використовувати стандартними засобами роботи з XML. Для використання XML визначено спеціальну мову. Приклад опису конфігурації одного з пристроїв введення параметра показаний на рис. 11.

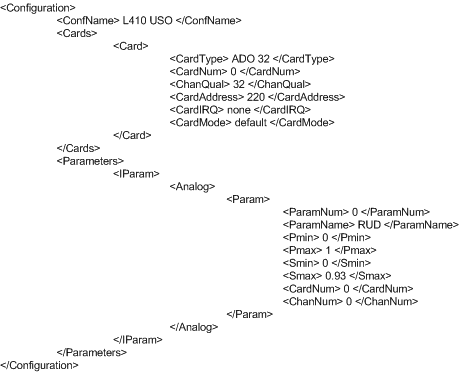


Рис. 11. Приклад опису конфігурації пристрою введення параметра

Для відновлення пульта інструктора тренажера пропонується підхід на основі ролей. Вводиться поняття авіаційної ролі AR = ‹R, I›, де R – множина ролевих видів; I =<T, F> – множина, яка містить елементи користувацького інтерфейсу, кожен має дві складових T – опис типу інформації та F – опис форми її представлення (індикатор, шкала-стрілка, транспарант, візир, цифрове табло, тумблер). Між R та Iіснують відповідні ієрархічні залежності. AR є основою для побудови об’єктно-оріентованого середовища програмування користувацького інтерфейсу (рис. 12).

Для реалізації запропонованого підходу розроблена архітектура засобів, які дозволяють створення та виконання сценаріїв польоту, створення видів та обмін даними пульта з іншими компонентами тренажера (рис. 13).

Сховища конфігурацій видів та сценаріїв польотів забезпечують зберігання даних про задані елементи пульта в XML-форматі. Розглянуто обґрунтування адекватності переробленого програмного забезпечення.



Рис. 12. Фрагмент середовища програмування інтерфейсу

Для перевірки адекватності функціонування переробленого програмного забезпечення реальному об'єкту використовувалися кількісні та якісні оцінки.Кількісна оцінка здійснювалася такими шляхами:

* точковим – порівняння по методиці розробника тренажера значень обчислених експлуатаційних параметрів, з точками реперів, що наведені в технічному описі тренажера;
* інтервальним – порівняння розрахованих характеристик з характеристиками, що наведені в документах реального літака (керівництво по льотній експлуатації, керівництво по технічній експлуатації, технічний опис двигуна та його систем, технічний опис авіаційного та радіоелектронного устаткування літака, дані бортової системи реєстрації польотної інформації).



Рис. 13. Архітектура засобів для реалізації пульта інструктора

Якісна оцінка поведінки тренажера на різних режимах та етапах польоту здійснювалась шляхом експертного оцінювання лінійними пілотами-експертами.

Результати застосування точкової оцінки наведені в таблиці, а результати інтервальної оцінки – на рис. 14, 15.

*Таблиця*

**Точкові оцінювання адекватності тренажера TL-410M**

|  |  |  |  |
| --- | --- | --- | --- |
| Етап польоту | Контрольований параметр | Контрольне значення | Розраховане значення |
| Розгін | Час з моменту розблокування колодки до досягнення швидкості *V* = 150 км/ г | *t* = 14 ± 2 с | 14.7 с |
| Швидкість підйому | Вертикальна швидкість підйому літака | *Vy*= 10 ± 1.5 м/с | 10.72 м/с |
| Льотні характеристики | Путня швидкість, що крутить момент, тангаж літака | *V* = 250 ± 25 км/г; *Мк*= 49 ± 5 %;  = 4.50 | 258.7 км/г  45.7%; 4.190 |
| Розгін у польоті | Час збільшення путньої швидкості літака *V* = 200 км/г, *V* = 300 км/г | *t* = 32 ± 4 с | 31.44 с |
| Гальмування у польоті | Час зниження путньої швидкості з  *V* = 300 км/г до *V* = 200 км/г | *t* = 29 ± 4 с | 31.14 с |



Рис. 14 Потужність двигуна Рис. 15 Швидкість польоту

Для реалізації статистичного підходу було застосовано методику, яка на прикладі оцінки адекватності окремого параметра, у вибраний момент часу, при виконанні заданого етапу польоту має наступні кроки:

1. на тренажері виконується статистичний експеримент по моделюванню заданої польотної ситуації та для вибраного параметра обчислюється модельна оцінка дисперсії  ;
2. виконується перший реальний політ та фіксується  (траєкторія зміни вибраного параметра в часі);
3. витримуючи умови реального польоту, дослідник моделює на тренажері аналогічну траєкторію ;
4. для вибраного перетину часу  обчислюється нев'язність  яка розглядається як перший елемент вибірки випадкової величини ;
5. оскільки випадкова величина  утворена поєднанням великого числа різних випадкових складових (залишків, нев'язності, похибок), то згідно центральній граничній теоремі теорії вірогідності, розподіл випадкової величини підкоряється нормальному закону:

,

де *а* – показник адекватності (спочатку *а* = 0,5).

Тоді апостеріорний розподіл адекватності з урахуванням отриманої інформації  представляється формулою:

,

де  – апріорний закон розподілу показника адекватності (спочатку рівномірний закон розподілу на інтервалі [0, 1]);  –постійний коефіцієнт пропорційності; ;

1. виконується другий реальний політ, фіксується  (чергова траєкторія зміни вибраного параметра), а на тренажері відтворюється  модельна траєкторія того ж параметра та обчислюється  наступний елемент вибірки випадкової величини  ;
2. як апріорний розподільник  використовується апостеріорний розподіл , отриманий на попередньому кроці. Можна показати, що новий апостеріорний розподіл (з урахуванням інформації  та ) приймає вигляд:

,

де , і далі 

де  – коефіцієнт пропорційності;  та .

1. Після багатократного повторення кроків (2 – 7) апостеріорний розподіл адекватності наближається до стабільної форми, а його мода визначає дійсне значення адекватності об'єкту (літака) та ПЗ (тренажера) по досліджуваному параметру (рис. 16).



Рис. 16. Результати оцінки адекватності моделі по параметру .

Основні наукові та практичні результати роботи полягають у наступному:

1. Проведено аналіз задачі реінженерії програмного забезпечення, в цілому, та зокрема реінженерії програмного забезпечення інформаційно-моделюючих комплексів тренажерного типу. Обґрунтовано актуальність вирішення цієї задачі для реінженерії програмного забезпечення цифрових авіаційних тренажерів шляхом застосування характеристик та властивостей об'єкту моделювання;
2. Вперше розроблено метод керованої об’єктом реінженерії програмного забезпечення, який на відміну від існуючих методів, враховує особливості інформаційно-моделюючих тренажерних комплексів та забезпечує реінженерію програмного забезпечення авіаційних тренажерів при заміні обчислювального обладнання та відсутності точного опису динамічної моделі домена. Застосування методу дозволяє відновлювати працездатність успадкованого програмного забезпечення авіаційних тренажерів, продовжуючи термін їх використання;
3. Вперше, на прикладі успадкованого програмного забезпечення авіаційного тренажеру ТЛ410М розроблено схеми реінженерії програмного забезпечення, які враховують особливості побудови успадкованого програмного забезпечення, що пов’язані з архітектурою технічних засобів, специфічними прийомами розробників, неповним документуванням. Схеми є узагальненим представленням процесів реінженерії та можуть бути застосовані для виконання реінженерії програмного забезпечення в інших доменах;
4. Розроблено і реалізовано інструменти, які забезпечують розуміння успадкованого програмного забезпечення (на основі схеми «екстрактор-абстрактор») та отримання інформації про обладнання тренажеру (вимірювачі, генератори тестових сигналів) необхідні при реінженерії програмного забезпечення;
5. Метод реалізовано шляхом виконання реверсивної інженерії та розробкою нового програмного забезпечення, частково, з урахуванням результату реверсу або заново;
6. Розроблено та реалізовано архітектуру засобів реінженерії програмного забезпечення авіаційних тренажерів, зокрема, шаблону для створення програмного забезпечення та середовища програмування для створення програмного забезпечення пультів інструктора. Шаблон та середовище побудовано з використанням XML, шляхом розробки спеціальних мов та протоколів;
7. Проведено практичну реінженерію програмного забезпечення технічного комплексу авіаційного тренажеру ТЛ-410М, та експеримент за розробленими у дисертації методом для обґрунтування адекватності функціонування переробленого програмного забезпечення функціонуванню реального об'єкту.

СПИСОК ОПУБЛІКОВАНИХ АВТОРОМ ПРАЦЬ

1. Хоменко В.А. Шаблон программного обеспечения устройств связи с объектом авиационных тренажеров / Хоменко В.А., Сидоров Е.Н., Мендзебровский И.Б. // Проблеми програмування; HAH України.- 2008.- №2, -3.- С. 239-249.
2. Сидоров Н.А. Реинженерия наследуемого програмного обеспечения авиационных тренажеров / Сидоров Н.А., Хоменко В.А., Недоводеев В.Т., Сидоров Е.Н. // Проблеми програмування; HAH України. - 2008.- №2, -3.- С. 288-299.
3. Сидоров Н.А. Реинженерия наследуемого программного обеспечения информационно-моделирующих тренажерных комплексов / Сидоров Н.А., Недоводеев В.Т., Сердюк И.П., Хоменко В.А., Сидоров Е.Н. //Управляющие системы и машины.-№4.-2008.-С.68-74.
4. Sidorov N. Reengineering of the Legacy Software: the air simulator case study.- //Sidorov N., Chomenko V., Sidorov E. //Proceedings of the third world Congress "Aviation in the XXI-ST century", "Safety in a aviation and space technology // Sept. 22-24, 2008.- Kyiv. Ukraine // V.- 2.-2008.-С. 33.88.-33.96".
5. Сидоров Е.Н. Метод управляемой объектом реинженерии программного обеспечения // Теоретичні та прикладні аспекти побудови програмних систем // Часть 1, - 22-26 сентябрь 2008 г. Киев, Черновцы.- С. 219-224.
6. Сидоров Е.Н. Метод метрического анализа гипертекста на основе Dexter модели // «VIII Міжнародна науково-технічна конференція» «АВІА - 2007».- 25-27 квітня. - т. 1., Київ.-2007.-С.13.149-13.152.
7. Сидоров Е.Н. Метрический анализ гипертекста// Матеріали конференції «Інженерія програмного забезпечення 2005». -К. -НАУ. -2005. -С.95-101
8. Сидоров Е.Н. Метод реінженерії успадкованого програмного забезпечення авіаційного тренажера// Тез. доп. Всеукр. конференції аспірантів і студентів « Інженерія програмного забезпечення 2007», Київ, 4-5 грудня 2007. -К.: НАУ, 2007. -С.26
9. Сидоров Е.Н. Подход к классификации метрических оценок гипертекста// Тез. доп. Всеукраїнської конференції аспірантів і студентів « Інженерія програмного забезпечення 2006», Конча-Заспа, 25-29 вересня, 2006р., К.: НАУ, 2007. –С.І8-22.
10. Сидоров Е.М. Архітектура програмного забезпечення пульту інструктора авіаційного тренажеру// Матеріали міжнародної конференції аспірантів і студентів «Інженерія програмного забезпечення 2008», Київ, 22-26 вересня 2008р., К.: НАУ, 2009. -С.47-52.

Підпис претендента Сидоров Є.М.