УДК 004.413:338.5

РЕИНЖЕНЕРИЯ НАСЛЕДУЕМОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ АВИАЦИОННЫХ ТРЕНАЖЕРОВ

Н.А. Сидоров, В.А. Хоменко, В.Т. Недоводеев, Е.Н. Сидоров

Национальный авиационный університет,
03058, проспект Комарова, 1.
Тел.: 406 7396, факс 497 8106,
e-mail: sna@nau.edu.ua

Рассмотрен метод управляемой объектом реинженерии наследуемого программного обеспечения, ориентированный на использование в области разработки информационно-моделирующих комплексов. Применение метода показано на примере реинженерии наследуемого программного обеспечения авиационного тренажера.

The object-driven software reengineering method is presented. The method application is demonstrated on the legacy software of the aviation simulator.

Существуют сложные наследуемые технические комплексы, состоящие из компонентов, часть которых активно используется, а другая часть утратила работоспособность вследствие морального и физического износа. Примером такого комплекса является авиационно-пилотажный комплекс «самолет-тренажер» [1]. Высокая стоимость компонентов, которые продолжают использоваться (самолет) делают актуальной задачу восстановления и поддержки работоспособности компонентов, утративших работоспособность (тренажер). Значительную часть таких комплексов, как правило, занимает программное обеспечение, которое вследствие замены морально устаревшего и физически изношенного аппаратного обеспечения требует соответствующей переработки. Методом переработки является реинженерия.

1. Реинженерия программного обеспечения

В общем виде реинженерия программного обеспечения требует выполнения процессов двух инженерий (рис. 1) – обратной и прямой [2]. На вход первого процесса, поступает наследуемое программное обеспечение и дополнительная информация о домене. В результате выполнения процесса строится модель – представление о домене. Данная модель, затем используется в процессах прямой инженерии для создания нового программного обеспечения.

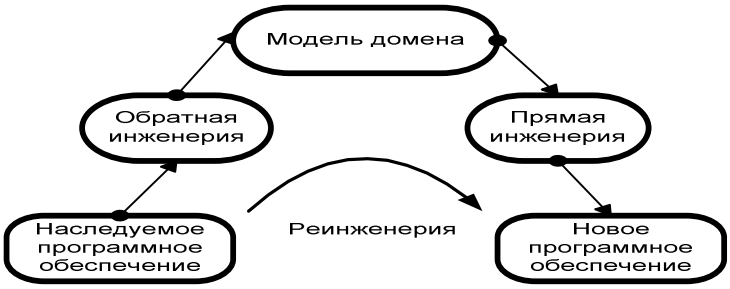

Рис. 1. Процессы реинженерии

Домен, в контексте инженерии программного обеспечения – это прикладная область, для которой разрабатывается программное обеспечение [3]. Модель домена – это описание домена, которое строится путем выполнения одного из методов обратной инженерии – восстановление проектной информации [4] или доменный анализ [3]. Независимо от метода, для построения модели домена используется комбинация кода, существующей документации, опыта персонала, общих знаний о проблеме и прикладной области [4]. Представление доменной модели, в зависимости от решаемых задач и зрелости домена может быть в форме таксономии, функциональных моделей или доменного языка [4]. Если построение модели осуществляется первым из указанных методов обратной инженерии, то важную роль играет наследуемое программное обеспечение.

Реинженерия – это эффективный метод утилизации программного обеспечения – продления времени полезного применения наследуемого программного обеспечения, который может применятся для реализации утилизации в разных аспектах – восстановления, повторного использования и переработки программного обеспечения [5].

Восстановление программного обеспечения обычно выполняется в процессе его сопровождения. Реинженерия выступает как метод борьбы со старением программного обеспечения, которое характеризуется рядом симптомов [6]: «загрязнение» кода, утрата знаний о программном обеспечении, плохой лексикон (стиль) [7], ослабление сцепления компонентов, «расслоение» архитектуры. Причины возникновения симптомов устраняются в действующем программном обеспечении [8].

Подготовка к повторному использованию может выполнятся как в действующем программном обеспечении (на его репликации), так и на ликвидируемом. Обычно повторному использованию подвергаются отдельные компоненты программного обеспечения, которые предварительно перерабатываются путем применения реинженерии – вследствие изменения их функциональности или вследствие миграции (новые ЭВМ, операционная система, язык программирования).

Переработка программного обеспечения при утилизации выполняется тогда, когда осуществляется миграция программного обеспечения. Особенно «тяжелым» является тот случай, который связан с миграцией наследуемого программного обеспечений на новую аппаратную платформу. Задача миграции возникает вследствие морального старения и физического износа аппаратной платформы. Это ведет к тому, что вычислительный комплекс не может больше использоваться, а наследуемое программное обеспечение или его отдельные работающие части нельзя выполнить, так как отсутствует вычислитель. В этом случае, как правило, меняются и операционная система и язык программирования.

Реализация процессов обратной инженерии связана с решением двух задач [9]: определение затрат, необходимых для построения модели домена; оценка качества процессов реверсивной инженерии и в целом реинженерии. Решение первой задачи зависит от зрелости домена и может основываться на моделях оценки стоимости программного обеспечения [10]. При этом, чем более зрелый домен, тем меньше затраты на осуществление процессов реверсивной инженерии. Качество процессов реверсивной инженерии обычно оценивают, показывая адекватность разработанного программного обеспечения наследуемому программному обеспечению или модели предметной области [9].

В работе предлагается метод, который разработан для применения на программном обеспечении, работающем в предметной области [11] и рассматриваются результаты его использования для переработки программного обеспечения авиационного тренажерного комплекса TL410M (самолет L 410), вызванной изменением аппаратной платформы. Вычислительный комплекс тренажера, построенный на базе ЭВМ Robotron, морально и физически устарел, поэтому требовал замены на современную технику. Более пятнадцати лет комплекс не эксплуатировался – стали неработоспособными ЭВМ, элементы устройства связи с объектом, значительная часть приборов в кабине пилота и на пульте инструктора. Вследствие этого стало невозможно выполнение наследуемого программного обеспечения тренажерного комплекса.

Оставшаяся часть работы состоит из трех частей. Во второй части представлен метод управляемой объектом реинженерии наследуемого программного обеспечения; в третьей – описано применение метода на примере реинженерии программного обеспечения авиационного тренажера; в четвертой рассматривается адекватность переработанного программного обеспечения.

2. Управляемая объектом реинженерия наследуемого программного обеспечения

В работе рассматривается метод реинженерии наследуемого программного обеспечения, которое создавалось для работы в предметной области. Это, так называемые, Е – программы [11]. Их использование направлено на автоматизацию деятельности человека или общества. При этом, программное обеспечение, становится частью реальной обстановки. Как правило, программное обеспечение рассматриваемого типа функционирует на гибридных (цифро-аналоговых) вычислительных комплексах, и значительная его часть связана с обработкой информации, циркулирующей между реальным объектом или его имитационной моделью и вычислительным комплексом [12]. В состав таких комплексов входят аналого-цифровые и цифро-аналоговые преобразователи, датчики реального объекта или его модели.

Особенности предметной области и Е-программ определяют то, что реинженерия наследуемого программного обеспечения рассматриваемого типа кроме решения традиционных задач, связанных с восстановлением проектной информации, требует восстановления информации о реальном объекте или имитационной модели и особого подхода к решению задачи, доказательства адекватности функционирования построенного в результате реинженерии программного обеспечения поведению реального объекта или его имитационной модели.

Информация о реальном объекте, восстанавливаемая в процессе реверсивной инженерии, – это наборы входных параметров и их характеристик; характеристики датчиков, приборов и исполнительных устройств реального объекта или модели. При этом, информация, кроме использования традиционного пути ее получения – анализ наследуемого исходного кода и документации, должна получатся путем проведения экспериментальных исследований поведения реального объекта или модели.

Доказательство адекватности функционирования построенного программного обеспечения не может осуществляться традиционными способами – сравнением результатов выполнения наследуемого и вновь построенного программного обеспечения или сравнением результатов реверсивной инженерии с поведением

соответствующей модели предметной области [9]. Первый способ нельзя использовать, так как нет вычислительного оборудования, на котором можно выполнить наследуемое программное обеспечение, а второй, – так как модель предметной области, представленная в документации, как правило, содержит ошибки. Кроме этого, особое положение Е – программ в реальном мире указывает на то, что основное внимание при доказательстве адекватности должно уделяться подробному анализу поведения программы в реальных, определенных условиях функционирования [11]. Все это свидетельствует о том, что решающую роль в доказательстве адекватности функционирования разработанного в результате реинженерии программного обеспечения, функционированию реального объекта или имитационной модели должны играть характеристики и свойства реального объекта. Это составляет сущность предложенного метода реинженерии программного обеспечения (рис. 2).

Рис. 2. Схема метода реинженерии

3. Реинженерия наследуемого программного обеспечения авиационного тренажера TL 410M

Применение разработанного метода было выполнено на комплексном авиационном тренажере TL 410M (рис. 3).

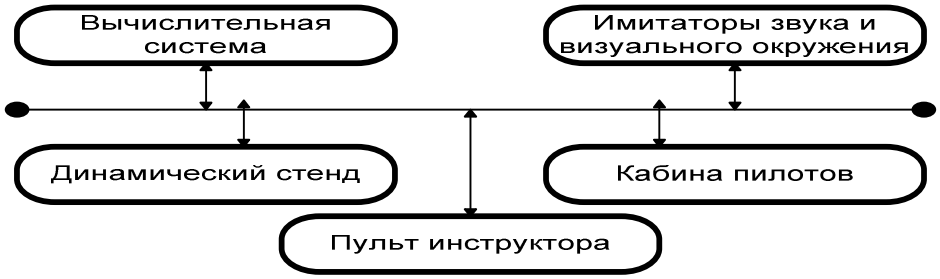

Рис. 3. Общая схема тренажера

Вычислительная система тренажера была построена на основе ЭВМ РОБОТРОН 4201 и аналого-цифрового устройства связи программной модели самолета с кабиной пилотов. Программное обеспечение написано на автокоде. Листинги наследуемого программного обеспечения представлены в документации, в составе семи альбомов, общим объемом около 32000 LOC. В тренажере была использована аналоговая имитация звукового окружения (белый шум и система фильтров), которая на момент восстановления не работала. Для визуализации окружения был использован телевизионный имитатор на основе фронтальной монохромной проекционной системы и стационарного планшета на рабочем месте инструктора (координаторы, макет аэродрома). Динамический стенд не работал и восстановлению не подлежал. Пульт инструктора содержал приборы, дублирующие те, что находились в кабине пилотов и телевизионный приемник, на вход которого подавался такой же сигнал, какой отображал визуальное окружение в кабине. Кабина пилотов имитировала кабину реального самолета L410.

На рис. 4 показана общая схема выполненной миграции аппаратного и программного обеспечения тренажера. Основной вычислитель (ЭВМ Robotron) и устройство связи с объектом были заменены на промышленный компьютер. Телевизионная система визуализации была заменена на компьютерную, на базе персонального компьютера и проектора. Для организации вычислений на промышленном компьютере была использована операционная система MS DOS, а на персональном компьютере – Windows. Миграция наследуемого программного обеспечения выполнена из языка ассемблерного типа в язык высокого уровня С. Кроме этого, были выполнены работы по восстановлению электротехнической схемы тренажера, которая

обеспечивала связь имитационной модели с кабиной самолета (кабельная система, электропитание, имитаторы функциональных узлов и коммуникационной аппаратуры, электрооборудование).

Так как, построенное программное обеспечение нельзя было проверить на правильность функционирования путем выполнения наследуемого программного обеспечения (отсутствовал вычислитель), то реверсивная инженерия, кроме традиционных процессов, включала процесс дополнительной взаимной проверки наследуемого кода и модели.

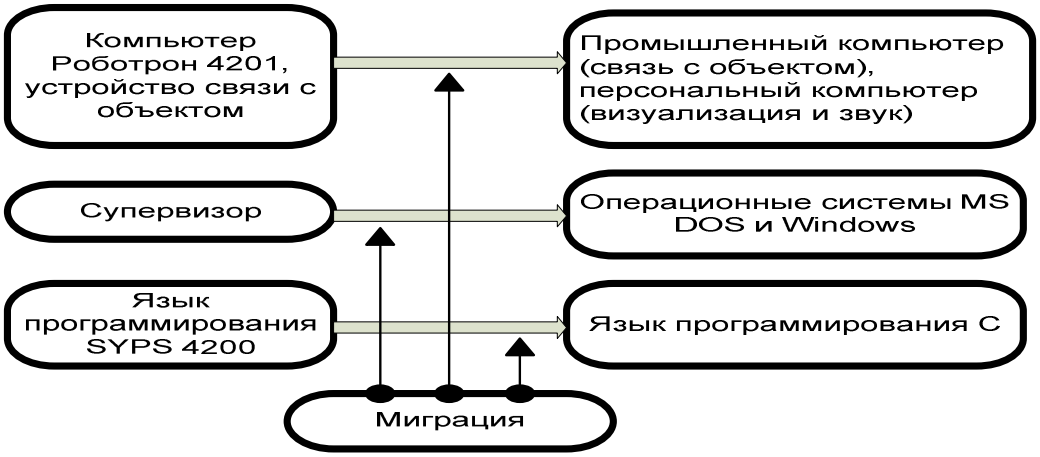

Рис. 4. Миграция аппаратного и программного обеспечения тренажера

Проверить математическую модель домена и наследуемый исходный код было необходимо, вследствие таких причин:

- ограниченной информации о принципах моделирования динамики полета самолета и его систем, которые были использованы разработчиком во время создания тренажера (1972);
- наличие ошибок случайно или умышленно внесенных в описания модели и исходный код, представленные в технической документации.

Проверка была выполнена в следующих аспектах:

– масштабирования переменных – осуществлялся пересчет коэффициентов уравнений математических моделей имитаторов (сравнение выражений приведенных в документации и их интерпретация в коде листингов показало существенные различия);

– реализации коэффициентов – использованы разработчиками наследуемого программного обеспечения в связи с ограниченными характеристиками компьютера моделирования;

– реализации методов решения математических задач.

Больше всего ошибок было в коэффициентах уравнений модели, описывающей динамику полета. Поэтому для уточнения коэффициентов использовался наследуемый исходный код, в котором определялись части, осуществляющие вычисление коэффициентов. Интерпретация этих частей «за столом» позволила определить значения ошибочных коэффициентов. Чтобы показать характер ошибок, допущенных случайно или умышленно в математической модели, рассмотрим пример.

В математической модели силовой установки аналоговый параметр n_v – обороты воздушного винта представлен модельным уравнением вида:

$$n_v = \frac{k}{1+\tau p}, \qquad (1)$$

где k – коэффициент усиления; τ – постоянная времени винта; p – оператор Лапласа.

После преобразования выражение (1) записывается в виде дифференциального уравнения:

$$\frac{dn_v}{dt} = \frac{k - n_v}{\tau} = f(n_v). \qquad (2)$$

В соответствии с алгоритмом, принятым в процедуре моделирования, уравнение (2) должно интегрироваться методом Эйлера:

$$n_{v(i+1)} = n_v + h \cdot f(n_{v(i)}), \qquad (3)$$

где h – шаг интегрирования (при расчете параметра принято соглашение – шаг интегрирования равняется шагу моделирования).

В исходной программе процедуру решения уравнения (2) представляет следующий текст:

```
16334 01 02 0416      S27      LDA
16335 100400                   UVR
16336 140040                   LOA
16337 0405 76                  KAR 2
16340 00 04 0101               SPA AA2
16341 0405 75                  KAR 3
16342 140401                   EKA
16343 00 06 0101               ADD AA2
16344 00 06 1757               ADD K12                    (4)
16345 01 07 0512               SUB NVP”
16346 0405 75                  KAR 3
16347 01 06 0512               ADD NVP”
…
16325 01 04 0512               SPA  NVP”
```

где – DEV – значение, указывают положение рычага управления воздушным винтом; К12 = 0,595 – константа.

Сравнение (3) и результата интерпретации (4) показывает, что в тексте (4) отсутствует шаг интегрирования h (при моделировании работы двигателей он равняется 0,12 с). Ошибка біла обнаружена путем проведения расчетов „за столом” – полученный переходный процесс $n_v = n_v(t)$ не соответствовал физическому смыслу. Оказалось, что такая ошибка присутствует при вычислении и других параметров силовой установки, где переходные процессы рассчитываются методом Эйлера.

Наследуемое программное обеспечение (рис. 5) характеризует дискретный процесс моделирования с периодом дискретизации 60 мс. За это время вычислитель успевал рассчитать значения параметров всех четырех имитаторов и завершить обменные операции. При этом необходимое быстродействие достигалось за счет следующего:

– упрощения математических моделей динамики полета и систем самолета;
– заменой аналитического описания реальных (динамических) процессов зависимостями в виде таблиц решений;
– использованием простых, но быстрых методов интерполяции функций;
– использованием простого метода интегрирования дифференциальных уравнений (метод Эйлера).

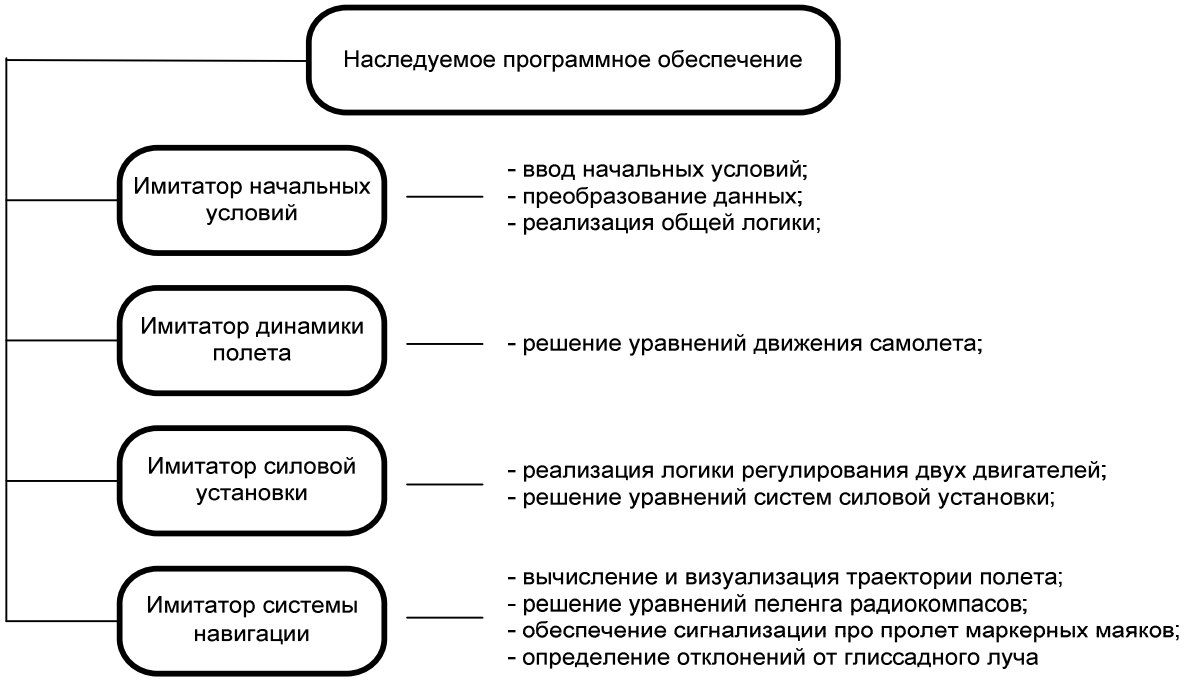

Рис. 5. Состав наследуемого программного обеспечения

Реинженерия наследуемого программного обеспечения после проверки моделей исходного кода осуществлялась в два этапа: реверсивная инженерия – построение высокоуровневого алгоритмического представления; прямая инженерия – по полученным алгоритмам строился код нового программного

обеспечения в языке С. Для выполнения реверсивной инженерии был построен специальный инструмент – абстрактор [13, 14].

Результат работы абстрактора и прямой инженерии на фрагменте кода показан на рис. 6. Рассматривается модель параметра $fN1 = fJMO(fTLS + fN1^* \cdot fTN1)$ – признака пребывания двигателя в режиме раскрутки ротора стартером:

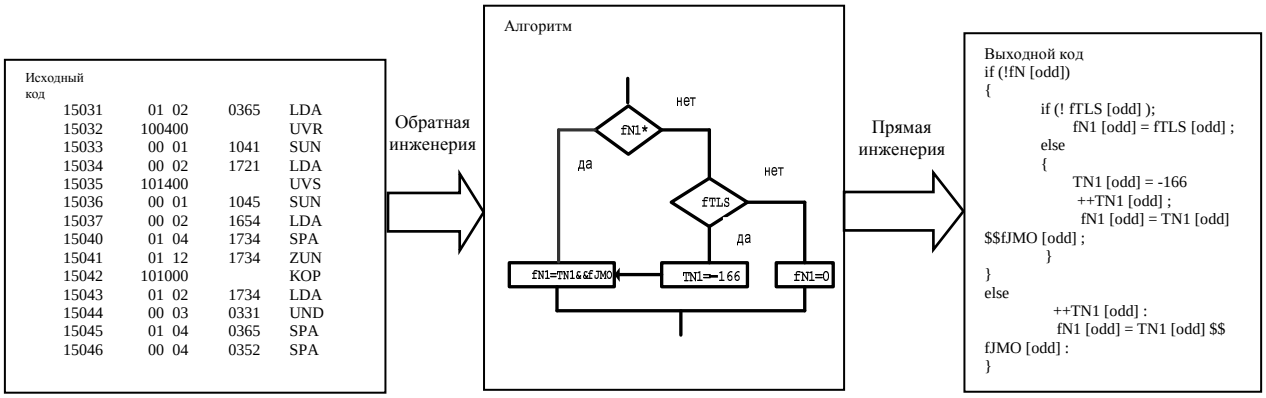

Рис. 6. Результат реинженерии (фрагмент)

Вследствие полной замены вычислительного комплекса часть программного обеспечения была создана заново. Основу его составляет система информационного обмена с объектом.

3.1. Программное обеспечение системы информационного обмена с объектом. Состав и структура нового технического обеспечения вычислительной системы тренажера определялась следующими факторами:

- объект управления – тренажер – система реального времени, для которой должно обеспечиваться заданное допустимое время отзыва;
- вычислительная система тренажера должна обеспечивать ввод и вывод большого количества входных и исходных параметров (около 120);
- визуализация полетного окружения тренажера должна быть реализована на высокоскоростном вычислителе со специальными характеристиками видеоподсистемы.

Эти факторы определили распределенную архитектуру вычислительной системы тренажера с несколькими специализированными вычислителями и периферийными устройствами (рис. 7), которая включает три вычислителя: промышленный компьютер для вычисления моделей имитаторов и ввода-вывода данных; компьютер для реализации имитаторов визуального окружения и шума; компьютер для пульта инструктора.

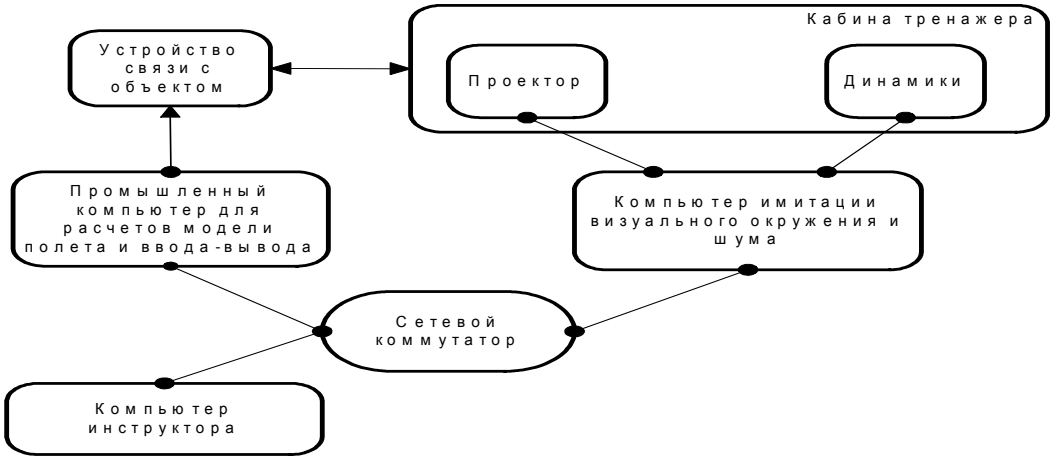

Рис. 7. Архитектура технического обеспечения моделирующей системы тренажеру

Обмен данными между вычислителями организован через локальную компьютерную сеть на основе технологии Ethernet по протоколу TCP/IP. Техническое обеспечение вычислительной системы включает устройства связи с объектом, компьютерный проектор и усилитель звука с громкоговорителями. Устройства связи с объектом относятся к одному из следующих типов: аналого-цифровые преобразователи (АЦП); цифро-аналоговые преобразователи (ЦАП); устройства ввода-вывода логических сигналов. Все устройства выполнены в виде плат, которые устанавливаются в промышленный компьютер на шину обмена типа ISA.

3.2. Программное обеспечение устройства связи с объектом. Программное обеспечение устройства связи с объектом работает на промышленном компьютере, предназначено для обеспечения обмена имитаторов (моделей) систем самолета входными и исходными параметрами с оборудованием кабины тренажера и строится путем настройки специально разработанного шаблона [15]. В задачу программного обеспечения входит следующее:

- конфигурирование устройств ввод-вывода промышленного компьютера;
- управление устройствами ввода-вывода;
- реализация обмена по протоколу TCP/IP;
- обеспечение стандартного интерфейса для обмена нормируемыми параметрами устройств ввода-вывода с моделями самолета.

Для обеспечения гибкости и масштабируемости подсистемы ввода-вывода была разработана двухуровневая модульная архитектура программного обеспечения (рис. 8), которая включает уровень драйверов устройств ввода-вывода и драйверов параметров. Функции драйверов обеспечивают управление операциями преобразования сигналов на входах устройств ввода-вывода и выдачу цифровых данных. Уровень драйверов обеспечивает конфигурирование устройств ввода-вывода при старте и управление ими во время обмена. Уровень драйверов параметров обеспечивает конфигурирование параметров, которые вводятся или выводятся из моделей систем самолета к объекту, и преобразования – цифровые значения сигналов устройств ввода-вывода – нормируемые аналоговые значения моделей.

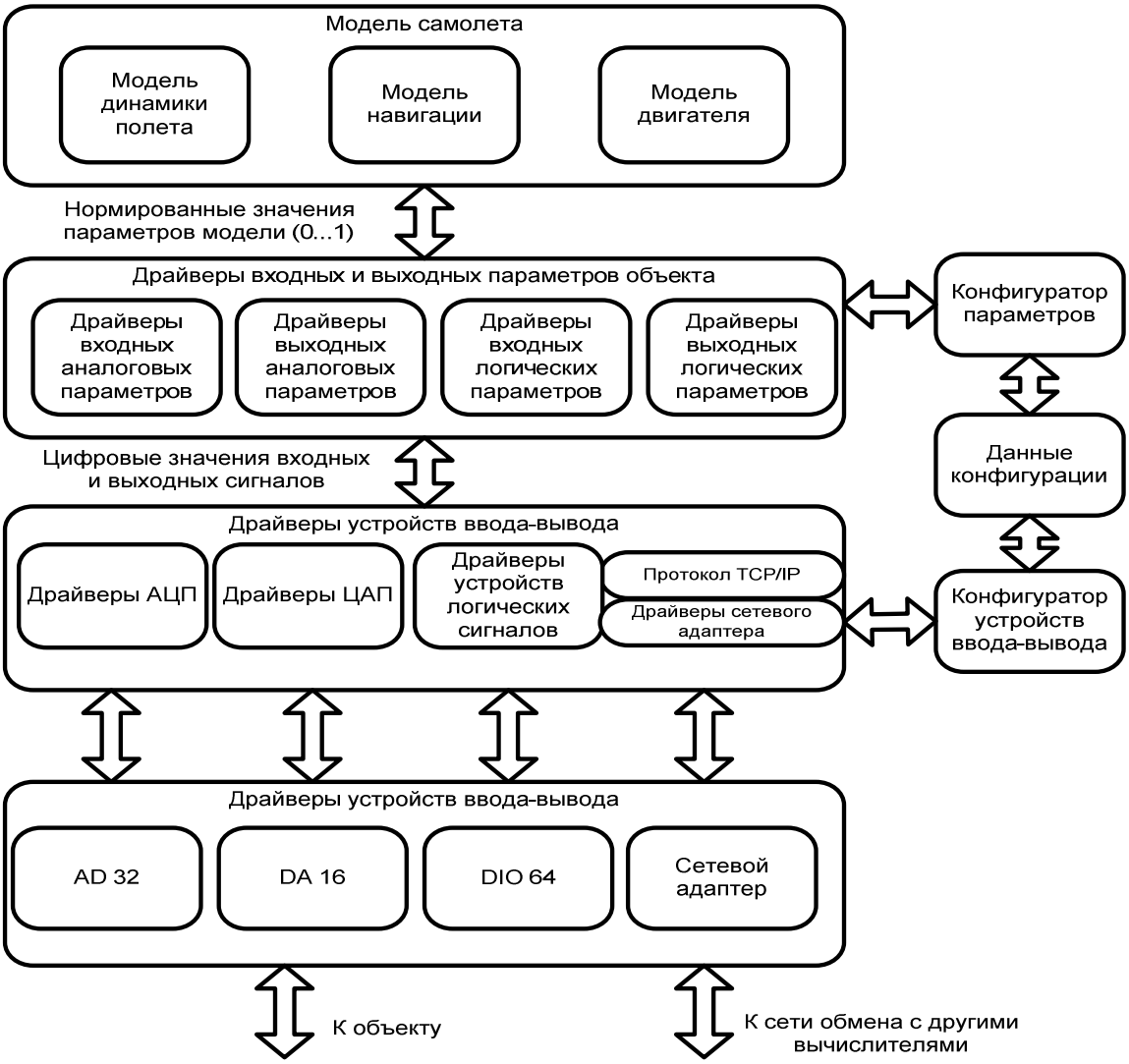

Рис. 8. Архитектура подсистемы связи с объектом

Конфигуратор устройств ввода-вывода осуществляет считывание данных конфигурации (номера плат, адреса, номера прерываний) для каждого из конфигурационных файлов, автоматическую настройку устройств и разовый контроль их функционирования.

Конфигуратор параметров осуществляет хранение конфигурации и считывание данных, необходимых для нормирования каждого параметра. Для каждого аналогового параметра ввода-вывода устанавливаются следующие конфигурационные настройки: номер параметра; минимальное и максимальное значение параметра и соответствующие им минимальное и максимальное значение напряжения сигналов устройств ввода-вывода.

3.3. Программное обеспечение имитаторов звукового и визуального окружения. Имитация звукового окружения пилотов в кабине тренажера обеспечивает слуховую информацию об имитируемых режимах руления, взлета, посадки и полета самолета. Звук в кабине самолета имеет сложный характер и формируется несколькими источниками шумов – двигателями и воздушными винтами, ветром снаружи кабины, оборудованием внутри кабины, шасси и другим оборудованием.

Параметры шумов зависят от режимов работы их источников и режимов полета самолета. Имитаторы шумов наследуемых тренажеров имеют аппаратную реализацию, основанную на смешении звуков, создаваемых несколькими специальными генераторами. Каждый такой генератор создает звук одного из источников шума самолета и меняет его характеристики по заданному закону в зависимости от параметров работы этого источника, подающиеся на имитатор от вычислителя тренажера. Как разновидность этого подхода (он был реализован в наследуемом тренажере) применяется получение шума различных источников с помощью генератора белого шума и набора регулируемых электрических фильтров.

Рассматривалось три подхода к восстановлению имитатора шума при реинженерии тренажера (рис. 9). Сущность первого заключается в использовании наследуемой реализации имитатора. Очевидно, что он может быть применен, если можно восстановить работоспособность наследуемого имитатора. В этом случае решается одна задача – стыковка имитатора с новой вычислительной системой тренажера. Сущность второго подхода заключается в разработке нового имитатора шума на основе информации о структуре и характеристиках старого. Такой подход требует наличия достаточной документации о наследуемом имитаторе или проведения обратной инженерии. Сущность третьего подхода заключается в разработке нового имитатора на основе информации, получаемой от объекта моделирования. Этот подход был реализован в процессе реинженерии. Он требует разработки собственной структуры имитатора, а также съема, анализа и формализации характеристик звуков самолета. Новый имитатор может быть реализован аппаратно – отдельным устройством или программно как приложение, функционирующее на одном из вычислителей восстановленного тренажера.

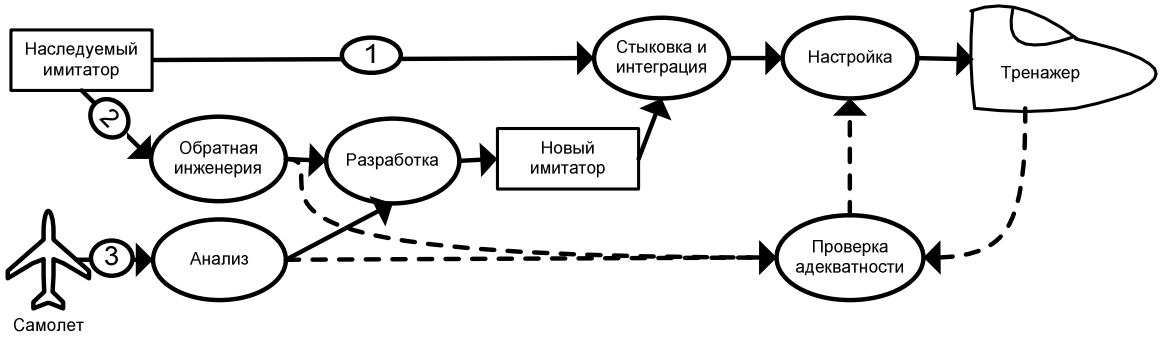

Рис. 9. Подходы к восстановлению имитатора шума

Любой из указанных подходов требует проверки адекватности звука, получаемого с помощью восстановленного имитатора, шуму самолета соответствующего типа. Для этого необходимо сравнивать его характеристики с характеристиками, снятыми с объекта (самолета), или полученными обратной инженерией от наследуемого имитатора (рис. 9).

Способ построения имитатора шума, который может применяться при реализации второго и третьего подходов – это использование вместо специальных генераторов, записанных цифровых отсчетов звука самолета, воспроизводимых проигрывателем с управляемыми параметрами (скорость проигрывания, громкость эффект эха и т.п.). Такой способ сочетает высокую скорость разработки имитатора, проводимую на основе базовых компонентов (проигрыватели, миксеры, эквалайзеры) и высокую реалистичность получаемого шума.

Имитатор визуальной обстановки при реинженерии подвергается практически полной замене [16]. Это связано с прогрессом, который претерпели технологии визуализации за последние два десятилетия. Первые имитаторы визуальной обстановки с цифровым синтезом изображений стандарта CGI появились в конце 70-х годов. В 90-х годах достигнутые качества и скорость работы цифровых синтезаторов изображений позволили полноценно использовать их в тренажерах. В восстановленном тренажере цифровой синтез изображений осуществляется на отдельном вычислителе, получающем от модели динамики полета параметры полета, необходимые для формирования изображения (координаты положения самолета, курс, крен, тангаж). Для интеграции имитатора визуальной обстановки в систему используется шаблон программного обеспечения устройств связи с объектом [15].

4. Адекватность переработанного программного обеспечения

Для проверки адекватности функционирования переработанного программного обеспечения реальному объекту использовались количественная и качественная оценки.

Количественная оценка осуществлялась такими путями:

- точечным – сравнение по методике разработчика тренажера значений вычисленных эксплуатационных параметров, с реперными точками; которые приведены в техническом описании тренажера ТЛ-410;

- интервальным – сравнение рассчитанных характеристик с характеристиками, которые приведены в документах реального самолета (руководство по летной эксплуатации, руководство по технической эксплуатации, техническое описание двигателя и его систем, техническое описание авиационного и радиоэлектронного оборудования самолета, данные бортовой системы регистрации полетной информации).

Качественная оценка поведения тренажера на разных режимах и этапах полета осуществляется путем экспертного оценивания линейными пилотами-экспертами.

Результаты применения точечной оценки представлены в таблице, а результаты интервальной оценки - на рис. 10, 11.

Таблица. Точечные оценивания адекватности тренажера TL 410M

Этап полета	Контролируемый параметр	Контрольное значение	Рассчитанное значение
Разгон	Время с момента разблокирования колодки до достижения скорости V = 150 км/ ч	t = 14 ± 2 с	14.7 с
Скороподъемность	Вертикальная скорость подъема самолета	V_y = 10 ± 1.5 м/с	10.72 м/с
Летные характеристики	Путевая скорость, крутящий момент, тангаж самолета	V = 250 ± 25 км/ч; $M_к$ = 49 ± 5 %; υ = 4.5^0	258.7 км/ч 45.7%; 4.19^0
Разгон в полете	Время увеличения путевой скорости самолета V = 200 км/ч, V = 300 км/ч	t = 32 ± 4 с	31.44 с
Торможение в полете	Время снижения путевой скорости с V = 300 км/ч до V = 200 км/ч	t = 29 ± 4 с	31.14 с

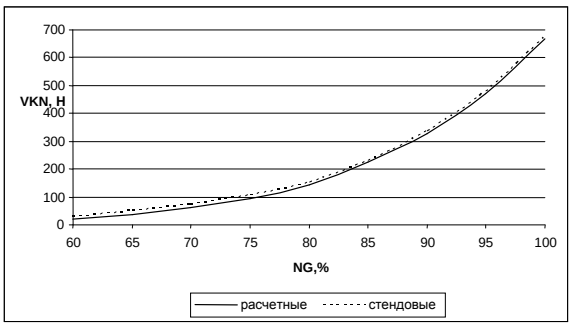

Рис. 10. Мощность двигателя (дроссельная характеристика)

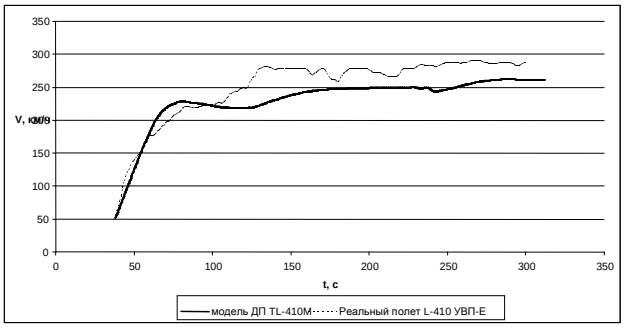

Рис. 11. Скорость полета (переходная характеристика)

Существуют два подхода к количественной оценке адекватности – детерминированный и статистический [17].

Результат проверки адекватности переработанного программного обеспечения на основе детерминированного подхода показали эффективность разработанного метода реинженерии наследуемого программного обеспечения.

Статистический подход к оценке адекватности планируется осуществить после проведения статистического эксперимента.

Сущность статистического подхода определяется следующим [17]:

– каждому из сравниваемых объектов (самолет и тренажер) свойственна статистическая неопределенность;

– объем регистрируемой в полете информации ограничен и поступает к исследователю нерегулярно;

– адекватность представляется как случайная величина, вероятное распределение которой формируется последовательно (по мере поступления полетной информации) на основании теории испытания статистических гипотез.

Для реализации статистического подхода разработана методика, которая на примере оценки адекватности отдельно взятого параметра в выбранный момент времени при выполнении заданного этапа полета имеет следующий вид:

1) на тренажере выполняется статистический эксперимент по моделированию заданной полетной ситуации и для выбранного параметра вычисляется модельная оценка дисперсии d^M;

2) выполняется первый реальный полет и фиксируется $x_1(t)$ – траектория изменения выбранного параметра во времени;

3) выдерживая условия реального полета, исследователь моделирует на тренажере аналогичную траекторию $y_1(t)$;

4) для выбранного сечения времени t^* вычисляется невязка $z_1 = x_1 - y_1$, которая рассматривается как первый элемент выборки случайной величины z;

5) так как случайная величина z образована сочетанием большого числа различных случайных составляющих (остатков, невязок, погрешностей), то согласно центральной предельной теореме теории вероятностей, распределение случайной величины z подчиняется нормальному закону:

$$g_z(z/a) = \frac{1}{\sqrt{2\pi d^M (1-a)/a}} \exp\left[-\frac{z^2}{2d^M(1-a)/a}\right]. \qquad (5)$$

Тогда апостериорное распределение адекватности с учетом полученной информации z_1 представляется формулой:

$$g_A(a/z_1) \sim f_A(a)\cdot g_z(z/a) = K_1\left(\frac{a}{1-a}\right)^{\alpha_1} \exp\left[-\frac{\beta_1 a}{1-a}\right], \qquad (6)$$

где K_1 – постоянный коэффициент пропорциональности; $\alpha_1 = 0{,}5$; $\beta_1 = 0{,}5(x_1 - y_1)^2 / d^M$;

1) выполняется второй реальный полет, фиксируется $x_2(t)$ – очередная траектория изменения выбранного параметра, а на тренажере воспроизводится $y_2(t)$ – модельная траектория того же параметра и вычисляется $z_2 = x_2 - y_2$ – следующий элемент выборки случайной величины z;

2) в качестве априорного распределителя $f_A(a)$ теперь используется апостериорное распределение (6), полученное на предыдущем шаге. Можно показать, что новое апостериорное распределение (с учетом информации z_1 и z_2) принимает вид

$$g_A(a/z_2) = K_2\left(\frac{a}{a-1}\right)^{\alpha_1+\alpha_2} \exp\left[-\frac{a}{1-a}(\beta_1+\beta_2)\right], \qquad (7)$$

где $K_2 = \text{const}$; $\alpha_2 = 0{,}5$; $\beta_2 = 0{,}5\cdot(x_2 - y_2)^2 / d^M$;

8. После многократного повторения пунктов (2–7) апостериорное распределение адекватности приближается к стабильной форме, а его мода определяет истинное значение адекватности объекта (самолета) и программного обеспечения (тренажера) по исследуемому параметру.

Заключение

Применение реинженерии программного обеспечения позволяет не только продлить использование устаревшего сложного научно-технического комплекса, но и осуществлять его совершенствующее сопровождение. Например, для пилотажно-моделирующего комплекса реинженерия обеспечивает следующее: возможность добавления новых функций; совершенствование моделей имитаторов; применение современных методов интегрирования.

1. *Сидоров М.О., Іванова Л.М., Хоменко В.А.* Методологічні принципи реінженерії програмного забезпечення успадкованих авіаційних тренажерів // Матеріали VIII Міжн. наук-техн. конф. АВІА-2007 – К. – 2007. – Т.1. – С. 13.119 – 13.122.
2. *Chikofsky E.J., Gross J.H.* Reverse Engineering and Design Recovery: Taxonomy. – IEEE Software. – Jan. 1990. – P. 13 –17.
3. *Prietto-Diaz R.*. Domain Analysis: An Introduction.-Software engineering Notes. – 1990. – V. 15, № 2. – P. 47–54.
4. *Biggerstaff T.J.* Design Recovery for Maintenance and Reuse. – Computer. – July. –1989. – P. 36 – 49.
5. *Сидоров Н.А.* Восстановление, переработка и повторное использование программного обеспечения. – УСиМ. – 1998. – N 4. – С. 71 – 79.
6. *Visaggio G.* Ageing of a Data Intensive Legacy System: Symptoms and Remedies. – J. Software Maintenance and Evolution. – 2001. – V. 13. – N 5. – P. 281 – 308.
7. *Сидоров Н.А.* Стилистика программного обеспечения // Проблемы программирования. – 2006. – № 2–3. – С. 245 – 255.
8. *Bianci A., Caivano D., Visaggio G.* Iterative Reengineering of Legacy Systems // IEEE Transactions of Software Engineering. – 2003. – V.29. – N.3. – P. 225 – 241.
9. *Rugaber S., Stirewale R.* Model-Driven Reverse Engineering // IEEE Software. – Jul/Ang. – 2004. – P. 45 –53.
10. *Сидоров Н.А., Бацепко Д.В., Василенко Ю.Н, Щебетин Ю.В.* Модели, методы и средства оценки стоимости программного обеспечения // Проблемы программирования. – 2006. – № 2–3. – К. – С. 290–299.
11. *Леман М.М.* Программы, жизненные циклы и законы эволюции программного обеспечения // ТИИЭР. – Proc.IEEE. – 1980. – Т. 68. – С. 26 – 46.
12. *Советов Б.Я., Яковлев С.А.* Моделирование систем: Учебн. для вузов – М.: – Высш. шк. – 2001. – 343 с.
13. *Сидоров Е.Н.* Метод реінженерії успадкованого програмного забезпечення авіаційного тренажеру // Тез. доп. Всеукраїнської конф. аспірантів і студентів «Інженерія програмного забезпечення 2007». – 2007. – С. 26.
14. *Маніцков М.К.* Зворотна інженерія, як основа технології відновлення роботи авіаційного тренажера. // Тез. доп. Всеукраїнської конф. аспірантів і студентів «Інженерія програмного забезпечення 2007». – К. – 2007. – С. 4.
15. *Хоменко В.А, .Сидоров Е.Н, Мендзебровский И.Б.* Шаблон программного обеспечения устройств связи с объектом авиационных тренажеров // Проблемы программирования. – 2008. – С. 23 – 46.
16. *Кузнецов С.В., Холод К.О.* Формирование матриц проекции для компьютерных генераторов изображения при использовании полупрофессиональных проекционных систем // Матеріали IV Міжнар. наук. конф. студентів та молодих учених. – 2004. – С. 35.
17. *Недоводеев В.Т.* Байсевская оценка адекватности модели полета: – Сб. науч. тр. – Киев: КИИГА, 1992. – С. 40–50.