ДОМЕННИЙ АНАЛІЗ

УДК 004.413:338.5

Ю.М.Рябокінь

ФОРМУВАННЯ ПОВТОРНО ВИКОРИСТОВУЄМИХ РІШЕНЬ ПРИ СТВОРЕННІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПУЛЬТА ІНСТРУКТОРА АВІАЦІЙНОГО ТРЕНАЖЕРУ МЕТОДОМ ДОМЕННОГО АНАЛІЗУ

Національний авіаційний університет
yulia.ryabokin@livenau.net

Навчання пілотів на авіаційних тренажерах управляється та контролюється інструктором. Основним інструментом інструктора є пульт – апаратно-програмна обчислювальна система з графічним інтерфейсом користувача. Для зниження витрат на розробку програмного забезпечення пульта пропонується застосування повторно використовуваних рішень, які формуються шляхом проведення доменного аналізу.

Обучение пилотов на авиационных тренажерах управляется и контролируется инструктором. Основным инструментом инструктора является пульт - аппаратно-программная вычислительная система с графическим интерфейсом пользователя. Для снижения затрат на разработку программного обеспечения пульта предлагается применение повторно используемых решений, которые формируются путем проведения доменного анализа.

Pilots training by air simulator is controlled and evaluated by instructor. The main tool of instructor is console that now is a hardware-software computer system with user interface. For reduction of the costs on software development of the console is proposed use reusable solution, that are formed by domain analysis.

Ключові слова: авіаційний тренажер, пульт інструктора, повторне використання, доменний аналіз, бібліотека класів, доменно-специфічна мова.

Вступ

Питання забезпечення безпеки є пріоритетними при організації повітряного руху і здійснення авіаперевезень пасажирів і вантажів. Практика експлуатації авіаційної техніки показує, що понад 80% причин всіх льотних подій і передумов до них відбуваються з вини членів екіпажу повітряного судна, тобто за рахунок так званого людського фактору. Все це визначає підвищені вимоги до рівня професійної підготовки кожного члену екіпажу повітряного судна, що особливо важливо під час прийняття рішення, як в штатних, так і в аварійних ситуаціях при пілотуванні повітряного судна, коли від швидкості і точності виконання операцій льотчика залежить безпека польоту. Якщо раніше, при підготовці спеціалістів по льотній експлуатації, в основному робили наголос на навчанні в реальних умовах польоту, то в теперішній час, в умовах економічної та екологічної кризи, для підготовки пілотів різних повітряних суден і для підтримки їх професійного рівня широке застосування отримати авіаційні тренажери [1].

Авіаційний тренажер є коштовним і складним комплексом обладнання. Значну частину цього комплексу складають обчислювальні пристрої та програмне забезпечення [2]. Програмне забезпечення призначене для імітації польоту різного роду літаків є надзвичайно складним та високорозвиненим продуктом. Для написання такого програмного забезпечення потрібні навички не лише програмістів, але й цілої низки інших фахівців авіаційної галузі. Аналізуючи ринок, можна зробити висновок, що основними розробниками програмного забезпечення у галузі авіації є самі авіакомпанії та конструкторські, авіабудівні підприємства. Що пов’язане в першу чергу із наявністю знань у спеціалістів компаній, як повинна функціонувати система.

В умовах обмеженого фінансування та труднощів пов’язаних з розробкою доменно-орієнтованого програмного забезпечення особливої актуальності набуває питання зниження витрат на створення програмного забезпечення, при збереженні якості та надійності програмного забезпечення та підвищення продуктивності програмного забезпечення. Одним із способів вирішення цієї проблеми є повторне використання при створенні програмного забезпечення. Тобто, використання раніше отриманих рішень-результатів: ідей, знань, вимог,

результатів аналізу, проектування, тестування та документування, компонентів коду під час розробки нового програмного забезпечення.

Аналіз останніх досліджень і публікацій

Аналіз сучасних дослідження говорить про те, що сучасні пульти інструктора (SLZ-242, Ми-8/Ми-17) реалізуються як апаратно-програмні системи на основі комп'ютерів, що включають засоби візуалізації інформації інтерактивної взаємодії з користувачем і інформаційного обміну з іншими компонентами тренажеру. Використання таких систем дозволяє відмовитися від дорогих спеціалізованих пристроїв відображення, вводу і фіксації інформації, що застосовувалися раніше (спеціалізовані індикатори, карти-планшети) і використати комп'ютерні і програмні компоненти [3]. Такі пульти інструктора складаються з одного чи декількох моніторів, миші, клавіатури та принтера. На моніторах за допомогою спеціалізованого програмного забезпечення імітується візуалізація інтерфейсу пульта інструктора (формується зображення індикаторів приладової дошки і відтворюється їх функціонування).

Знайти програмне забезпечення для створення пульта інструктора авіаційного тренажера з відкритими джерелами програмного коду майже не можливо, тому що розробники затратили на нього дуже багато часу та фінансів.

Як наслідок закритості такого плану розробок програмного забезпечення, є практична відсутність доступності широкому загалу інших типів програмного продукту (Microsoft Flight Simulator X і X-Plane), окрім спрямованих в першу чергу на ігроманів.

В літературі згадується лише 2 види спеціалізованого програмного забезпечення тренажерів [4], яке може використовуватися при створенні інтерфейсу пульта інструктора.

Система автоматизованого проектування Segambis дає можливість створювати динамічні образи різноманітних індикаторів та цілих приладових дощок.

Програма DeskSim (імітатор приладової дошки) дозволяє із заготовок (стрілок, шкал, табло) створювати макети приладових дощок, та легко змінювати розміщення приладів за певними рекомендаціями.

Проаналізовані системи для створення сучасних пультів інструкторів використовують підхід «фіксована функціональність», сутність якого полягає в тому, що склад і функціональність пульта визначаються на основі відомих типових задач, що виконуються інструктором при навчанні пілотів [5]. Основним недоліком такого підходу є те, що після створення пульта, види і форми інформації, що відображається та вводиться не змінюються. Для внесення змін необхідна наявність розробника. Тобто, кожен елемент, що міститься на пульті інструктора, розташовується у чітко визначеному місці і не може бути видалений у разі, якщо він не потрібний за даних умов (наприклад, при відпрацюванні певної польотної ситуації).

Закритість інформації про розробку програмного забезпечення пультів інструкторів авіаційних тренажерів не дає можливості вироблення рішень, що зможуть повторно використовуватись при створенні програмного забезпечення пультів інструкторів інших типів авіаційних тренажерів.

В роботі розглядається доменний аналіз пульта інструктора авіаційного тренажеру L-410 як метод отримання рішень, що будуть повторно використовуватись при створенні програмного забезпечення пультів інструкторів різних видів авіаційних тренажерів.

Постановка завдання

Основна мета даної роботи є формування повторно використовуваних рішень при створенні програмного забезпечення пульта інструктора авіаційного тренажеру. Для досягнення цієї мети в роботі пропонується використати метод доменного аналізу.

Повторне використання програмного забезпечення

Повторне використання програмного забезпечення – це використання компонентів існуючого програмного забезпечення для побудови нового [6].

Суть принципу повторного використання програмного забезпечення полягає в застосуванні в новій ситуації раніше отриманих результатів. Розглядаючи програмного забезпечення в широкому сенсі, можна вказати два рівня повторного використання результатів – ідей та знань; програмних продуктів та їх складових [7,8]. Тому повторно використовувані можуть бути не тільки компоненти коду, але й вимоги, результати аналізу, проектування, тестування та документування.

Повторне використання практикується відомими виробниками ПЗ, наприклад, AT&T, IBM, Ericsson, Hewlett-Packard, Motorola, Nec, Toshiba, розробившими власні технології повторного використання і створившими відповідні підрозділи для їх впровадження і

досліджується багатьма вченими, такими як Л. Бабенко, І. Вельбицький, М. Сидоров, V. Basili, T. Biggerstaff, G. Caldiera, E. Chisofsky, E. Horovitz, R. Pieto-Diaz, H. Rombach, I. Sommervile, P. Wegner.

Створення нового програмного забезпечення із застосуванням повторного використання складає два процеси (рис. 1). Суть першого полягає в розробці програмного забезпечення, а другого – в виробництві повторно використовуваних компонент. При цьому, перший процес ґрунтується на застосуванні повторно використовуваних компонент.

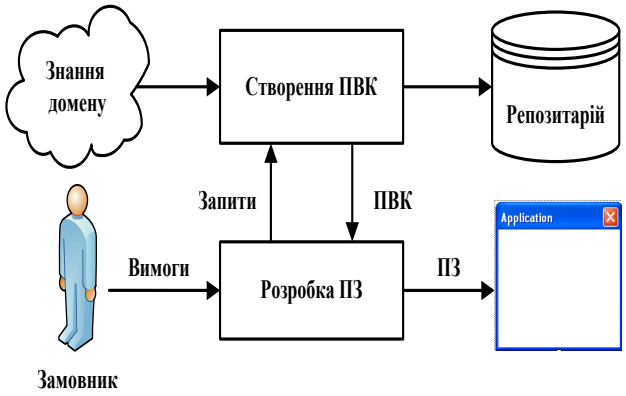

Рис.1. Розробка нового програмного забезпечення із застосуванням повторного використання

За способом реалізації процесів виробництва повторно використовуваних компонент повторне використання, що застосовується при розробці нового програмного забезпечення, може бути несистематичним та систематичним [9].

Несистематичне повторне використання – це використання компонентів існуючого (наслідуваного) програмного забезпечення при розробці нового програмного забезпечення. Цей тип повторного використання припускає використання елементів раніше розробленого програмного забезпечення шляхом їх виділення із наслідуваного програмного забезпечення, переробки та/чи настройки, та інтеграція в нове програмного забезпечення.

Суть систематичного повторного використання полягає в створенні «з нуля» компонентів, що реалізують функції для визначеного домену, і їх подальшого багатократного використання при розробці програмного забезпечення. Цей тип повторного використання передбачає наявність глибоких знань про домен. Основу систематичного повторного використання складає доменний аналіз.

Визначення доменного аналізу

Першою фазою доменної інженерії, що передує створенню систем є доменний аналіз. Він спрямований на ідентифікацію доменів, виявлення їх меж, дослідження схожості та відмінності систем.

Термін доменний аналіз вперше був введений Нейборсом [10] для позначення дій, що забезпечують виявлення об’єктів, операцій та відношень з метою їх повторного використання при розробці нових застосувань даного та інших доменів. Доменний аналіз безпосередньо пов’язаний з технікою набуття та використання знань. Виходячи з визначення в роботі [11], доменний аналіз розглядається як процес, в якому інформація, яка використовується при розробці програмного забезпечення визначається, збирається, структурується і організовується з метою її подальшого використання. Тобто, доменний аналіз забезпечує виявлення спільних об’єктів, операцій та відношень серед подібних застосувань в домені з метою їх повторного використання в розробці нових застосувань цього та інших доменів [12].

В даній роботі доменний аналіз використовується як фундаментальний крок при створенні реальних повторно використовуваних компонент. Доменний аналіз пов'язаний з повторним використанням, його метою є визначення інформації зв’язаною з доменом для подальшого використання при розробці застосувань, що відносяться до цієї області. Організації, що проводили доменний аналіз, який передував створенню повторно використовуваних компонент, показали значний успіх в повторному використанні [12]. Компоненти, що є результатами доменного аналізу краще придатні для повторного використання, оскільки збирають важливі функціональні вимоги домену; так, розробники швидко знаходять їх та легко застосовують при розробці нового програмного забезпечення.

Існують різні підходи до доменного аналізу, найбільш розповсюджені – Synthesis, Lubars, KAPTUR, Prieto-Diaz, FODA [13-14]. В статті пропонується скористатися фасетно-класифікаційним підходом R. Prieto-Diaz [12]. Особливістю даного підходу являється те, що Prieto-Diaz запропонував впровадити доменний аналіз в процес розробки програмного забезпечення, в якому продукти доменного аналізу безперервно переглядаються і уточнюються при створенні нових систем;

визначив конкретні під процеси (рис. 2) і проміжкові продукти, які передають глибоке розуміння процесу доменного аналізу; визначив входи і виходи під процесів доменного аналізу; описав процес доменного аналізу колекцією діаграм потоків даних, що відображають послідовність виконання доменного аналізу визначеної предметної області.

Отже, проведення доменного аналізу включає три послідовні під процеси – підготовка інформації, власно в аспекті авіаційної техніки, доменний аналіз, створення повторно використовуваних компонентів (рис. 3) [12].

Доменний аналіз здійснюється під керівництвом доменного аналітика, в задачі якого входить визначення входів процесів і результатів доменного аналізу за допомогою відповідних керівництв [12] по доменному аналізу.

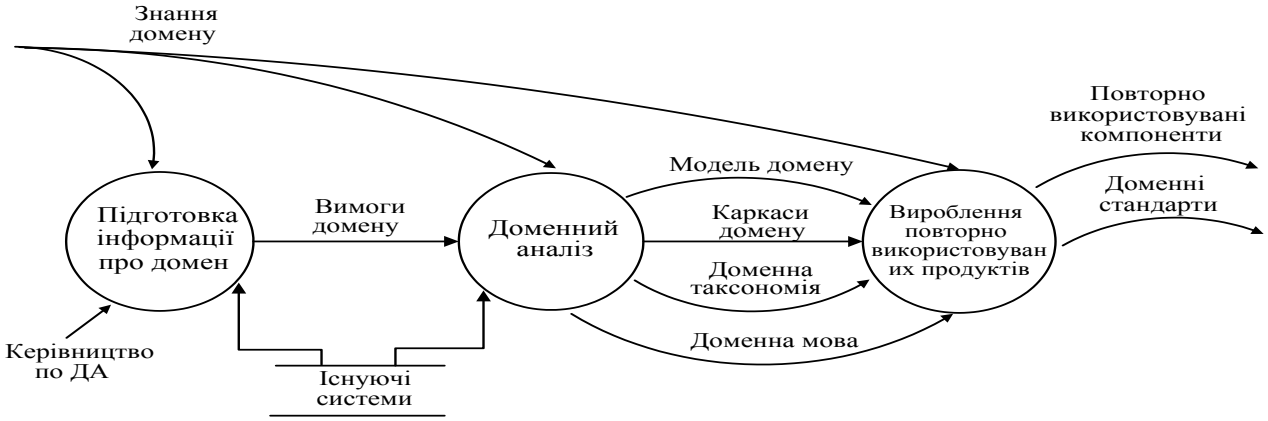

Рис.2. Під процеси доменного аналізу

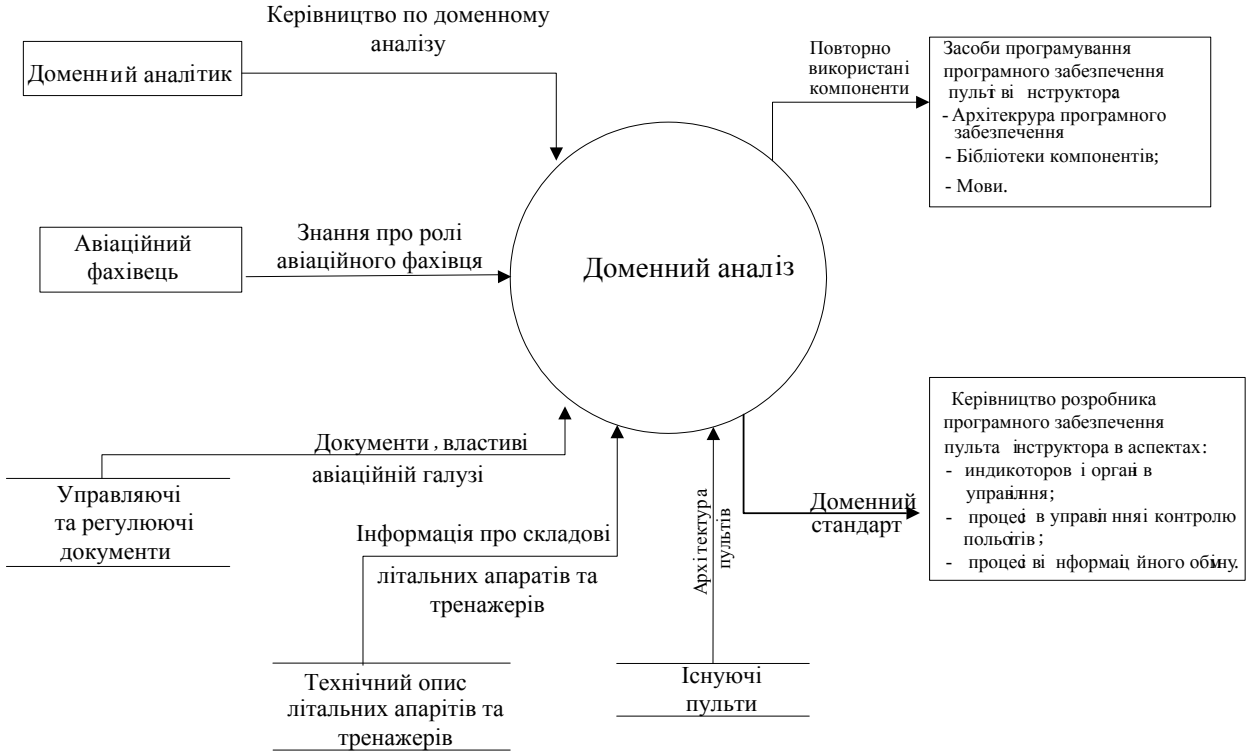

Рис. 3. Контекст доменного аналізу в авіаційному домені

Основними джерелами знань для доменного аналізу в даному контексті є:

1) авіаційні спеціалісти (інженер-конструктор авіаційної техніки, інструктори), що

володіють знаннями про складові пульту інструктора, їх конструкції;

2) управляючі і регулюючі документи – Конституція України [15], Повітряний кодекс України [16], документи ІСАО [17] (Керівництво по аеропортовим службам, Керівництво по запобіганню авіаційних подій, Керівництво по плануванню обслуговування повітряного руху), галузеві стандарти;

3) інформаційні документи – технічний опис літального апарату (наприклад, Авіаційний тренажер ТЛ-410. Технічний опис. Книга 1. Основні характеристики тренажера [18]), керівництво по льотній експлуатації [19];

4) існуючі пульти інструкторів.

В результаті доменного аналізу повинні бути вироблені колекції (бібліотеки) повторно використовуваних компонентів, такі як абстрактні класи, програмні шаблони, протоколи, що формують середовище реалізації доменної інженерії, що виконується з допомогою спеціальних середовищ програмного забезпечення, і доменний стандарт – керівництво розробника по використанню повторно використовуваних компонент і середовищ програмування для створення програмного забезпечення.

Обов'язковими продуктами доменного аналізу є вимоги до застосування домену, шаблони, таксономія домену, а необов’язковими є модель домену і мова домену. Загалом, доменний аналіз проводиться в два етапи – загальний аналіз домену і аналіз домену за окремими аспектами [11].

Загальний аналіз домену

Авіаційний тренажер є частиною пілотажно-навчального комплексу «літак-тренажер», який застосовується для навчання пілотів і представляє собою складну апаратно-програмну моделюючу систему, що імітує кабіну літака з органами управління та індикаторами, візуальне і звукове оточення пілотів, і поведінку літака при різноманітних фазах польоту [20].

Основне призначення авіаційного тренажеру:

1)навчання пілотуванню в звичайних та складних метеорологічних умовах;

2) навчання управлінню силовими установками, системами літака та обладнанням;

3) забезпечення виконання тренувального польоту за маршрутом із застосуванням радіонавігаційних засобів і засобів зв’язку;

4) забезпечення виконання зльоту і заходу на посадку і посадкових маневрів, тренувань дій екіпажу при виникненні в польоті різних нештатних ситуацій.

Навчання пілотів відбувається у відповідності з програмами підготовки (рис. 4), які формуються в залежності від конкретних задач навчання, і передбачають проведення серії учбових польотів при заданих умовах. Програми підготовки можуть змінюватися в період експлуатації літака на основі набутого досвіду пілотування. Тому, для ефективного виконування навчальних дій та швидкого аналізу даних, інструктору необхідно володіти детальною інформацією, представленою в доступному для огляду виді [21].

Одним із невід’ємних компонентів авіаційного тренажеру є пульт інструктора, який представляє собою інструмент для навчання пілотів. За допомогою пульта інструктор здійснює установку початкових умов польоту, контроль якості пілотування, імітує взаємодію пілотів з диспетчером, управляє імітацією відмов літака, розбирає помилки і особливості пілотування (рис. 5) [20].

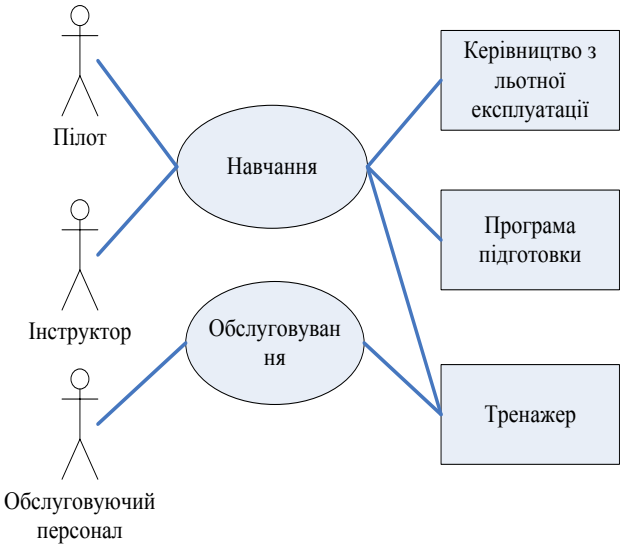

Рис. 4. Організація роботи тренажеру

Для оцінки якості пілотування і розбору польотів інструктор отримує значення відповідних параметрів з фіксацією їх на паперовому чи електронному носіях.

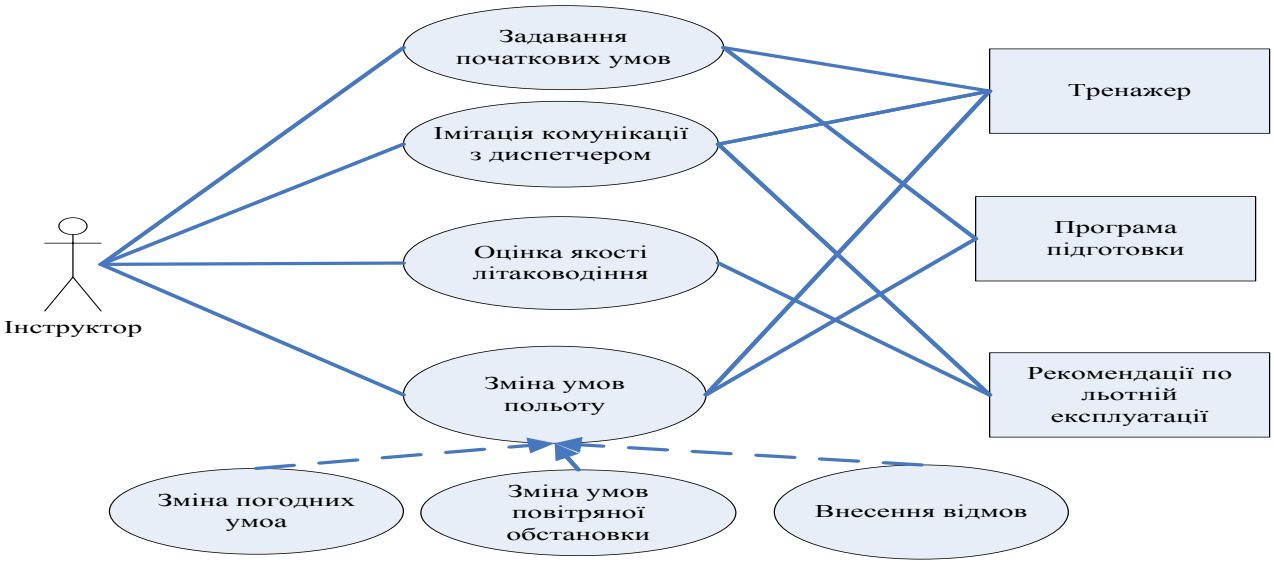

Рис. 5. Функції інструктора

Виходячи із загального аналізу процесу навчання пілотів, можна виділити основні аспекти частини прикладного домену «авіаційний тренажер» - пульта інструктора. До них відносяться: інтерфейсні елементи, процеси управління і контролю навчання, інформаційний обмін. В статті проводиться аналіз складової пульта інструктора – інтерфейсні елементи.

Доменний аналіз інтерфейсних елементів

Інтерфейс, через який інструктор отримує інформацію про політ і здійснює управління тренажером, називається користувальницьким інтерфейсом інструктора [22]. Інтерфейс інструктора становить безліч інтерфейсних елементів (елементи для відображення інформації та для управління процесом навчання), які призначені для полегшення роботи інструктора.

Відповідний фрагмент доменного аналізу інтерфейсних елементів, у вигляді діаграми потоків даних, представлено на рис. 6.

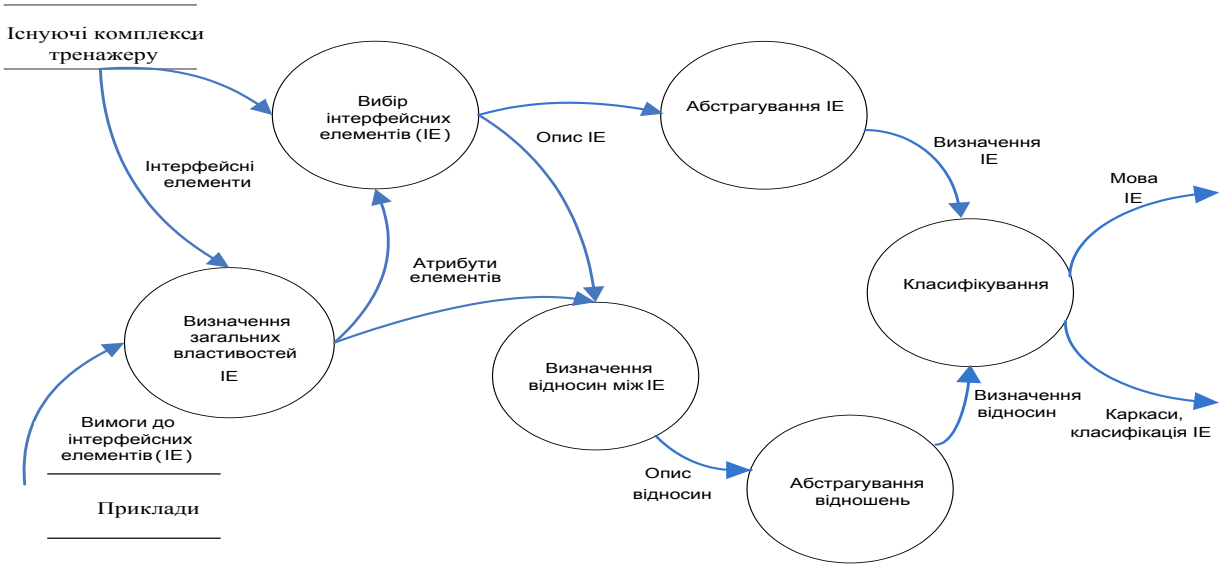

Рис. 6. Доменний аналіз інтерфейсних елементів

Використовуючи документацію по існуючим пультам інструктора і вимогам до інтерфейсних елементів (технічні специфікації), доменний аналітик виділяє спільні інтерфейсні елементи, властивості різноманітним типам авіаційних тренажерів їх функціональні властивості. На основі отриманої інформації здійснюється виділення інтерфейсних елементів (покажчик швидкості, висотомір, кнопки різноманітних відмов), їх опис.

Кожний інтерфейсний елемент представляється набором компонент – шкали, стрілки, діапазону, мітки, підпису та інш., які можна представити у вигляді класів (наприклад, клас CircularScale, клас CircularRange). Атрибути класів володіють загальними та індивідуальними властивостями (наприклад, загальні – колір, радіус, діапазон величин, а індивідуальні – довжина стрілки) [23–24]. Класи взаємодіють між собою (відношення композиції), визначаючи функціональність інтерфейсних елементів. Опис компонентів інтерфейсних елементів представляє собою спільний список компонентів і їх функцій.

Абстрагування – процес уявного виділення деяких елементів конкретної множини і відхилення їх від інших елементів цієї множини [25]. В результаті виконання цієї діяльності здійснюється виділення абстрактних класів індикаторів і органів управління та базового класу інтерфейсних елементів GadgetBase.

При побудові інтерфейсних елементів та настроюванні інтерфейсу під різні види задач пропонується узагальнення доменної інформації на основі поняття ролі. В авіаційному прикладному домені роль визначається, як сукупність функціональних обов'язків, властивих авіаційному фахівцю [23]. В процесі своєї діяльності авіаційний фахівець отримує і забезпечує визначену інформацію через технічні системи. Сукупність видів інформації і способів її відображення, властивій одній авіаційній ролі будемо називати рольовим представленням. Таку інформацію можна представити як набір окремих інформаційних елементів, які групуються в логічно і ергономічно пов'язані групи. Останні будемо називати панелями представлень. Наприклад, до панелей представлень інструктора можна віднести навігаційну приладову панель, панель управління силовою установкою, навколишнє візуальне оточення кабіни літака.

Користувальницький інтерфейс пульта інструктора, через який він отримує інформацію про політ і управляє тренажером, складають індикатори та органи управління [24]. Кожний індикатор відображає один параметр польоту. Множина індикаторів, доступних інструктору складає реалізацію його рольового представлення і повинна визначатися підмножиною всіх параметрів польоту, і способами їх представлення, прийнятими в авіації, з врахуванням конкретного типу літака. Органи управління пульта визначаються потребами управління тренажером.

Набір і вид інтерфейсних елементів індивідуальні для тренажера літака кожного типу, однак суттєво схожі. Кожний інтерфейсний елемент $ie = (it, pf)$ визначається двома складовими – видом інформації (параметром) it і формою її представлення pf. Рольове представлення rv можна описати як множину інтерфейсних елементів, що входять в нього $RV = \{ie_i \mid ie_i \in RV\}$, а роль R в свою чергу – як множину панелей представлення $R = \{RV \mid RV \in R\}$. Вся множина інтерфейсних елементів домену $IE = \{ie_i \mid ie_i \in IE, i = 1..d\}$ визначається як множина двійок $ie = (it, pf)$ всіх його підмножин IE_r, визначених для ролей: $R(IE) = \{IE_r \mid IE_r \subseteq IE\}$.

Інтерфейс для ролі інструктора синтезується шляхом вибору та настройки визначених інтерфейсних елементів: $r_{instr} = RV_{instr} \subseteq \{ie_i = (it, pf) \mid i = 1..l\}$.

Аналіз інтерфейсних елементів дає можливість виділити повторно використовувані рішення, які будуть застосовуватись при побудові інтерфейсів різних пультів інструкторів авіаційних тренажерів.

Результати доменного аналізу

В результаті проведення доменного аналізу та отримання відповідних знань домену можна здійснити класифікацію інтерфейсних елементів, шляхом їх групування за певними загальними ознаками. Результатом етапу класифікування є представлення класифікації інтерфейсних елементів їх каркасу та виділення доменно-специфічної мови – мови інтерфейсних елементів.

Класифікацію інтерфейсних елементів можна провести за наступними ознаками:

1) за призначенням (управляючі, призначені для введення інформації (перемикачі режимів, регулятори, кнопки введення значень) та індикатори, призначені для її відображення

(індикатори швидкості, висоти, відображення карт);

2) за способом відображення інформації (шкала-стрілка, транспарант, візир та інш.);

3) за способом введення інформації (аналогові, дискретні та цифрові).

Каркаси інтерфейсних елементів представляються у вигляді абстрактних класів, що знаходяться в ієрархії з базовим абстрактним класом «Інтерфейсний елемент» (рис.7).

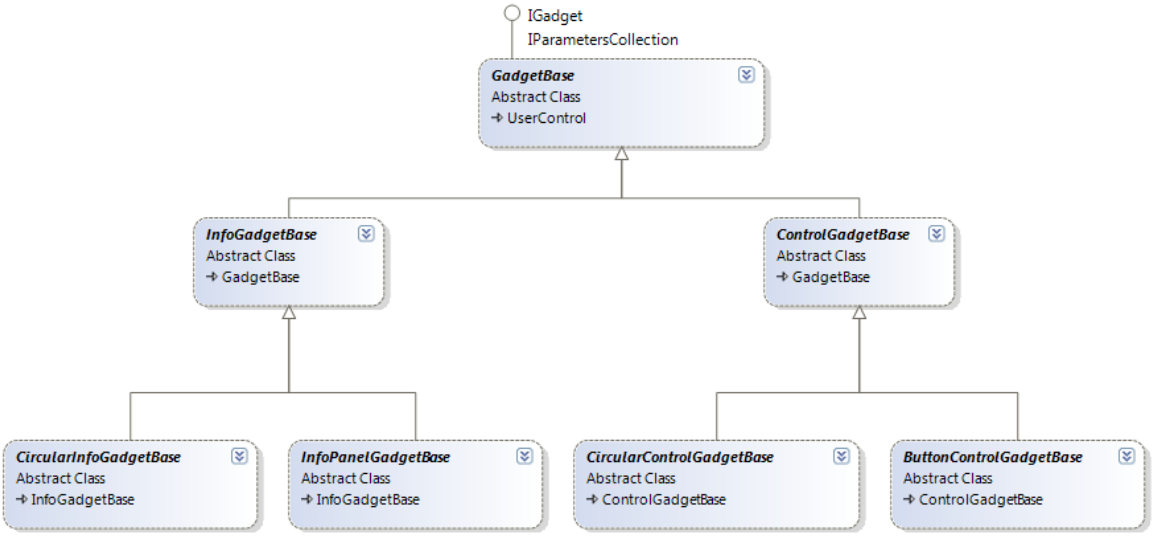

Рис. 7. Абстрактні класи інтерфейсних елементів

Класи індикаторів (InfoGadgetBase) та органів управління (ControlGadgetBase) наслідуються від базового класу всіх інтерфейсних елементів GadgetBase. InfoGadgetBase та ControlGadgetBase містять класи для реалізації елементів на основі шкал та стрілок (CircularGadgetBase, CircularControlBase), а також класи елементів на основі інформаційних табло та перемикачів (InfoPanelGadgetBase, ButtonControlGadgetBase).

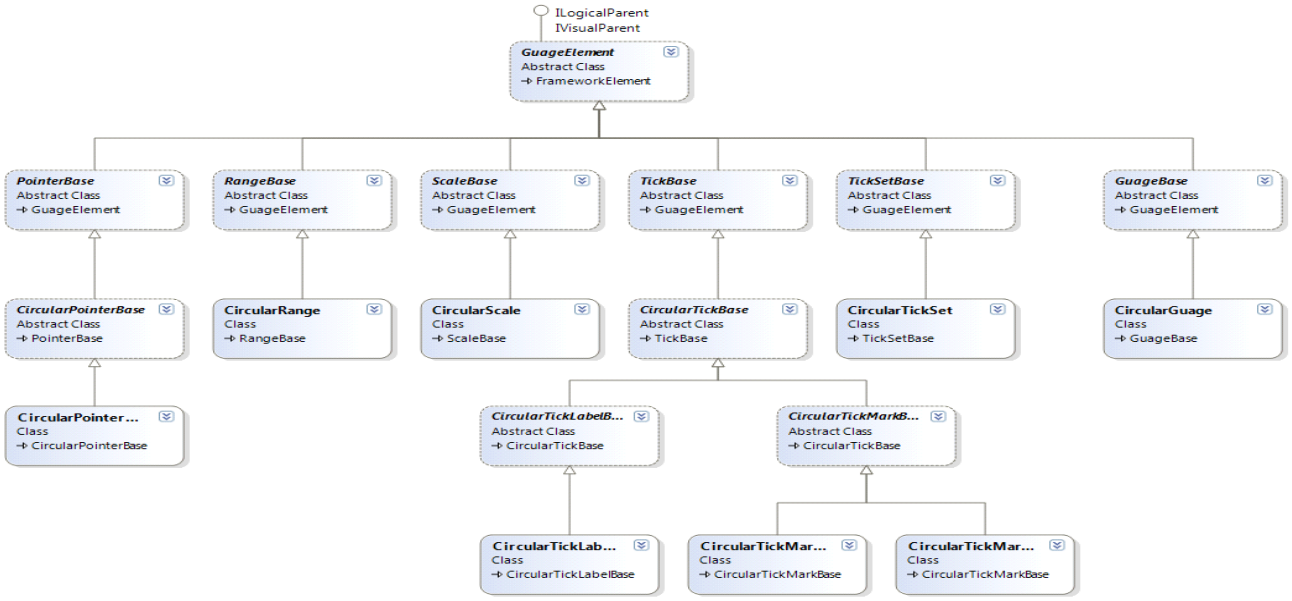

Рис. 7. Архітектура бібліотеки класів індикатору

В процесі доменного аналізу, на основі класифікації і специфікуванні властивостей елементів, побудована ієрархія класів, на базі якої розроблена бібліотека. Шаблон «компонувальник», використовується як базова архітектурна концепція ієрархії (рис. 7) [26].

Всі класи бібліотеки наслідуються від базового класу GaugeElement, який має можливість композиції собі подібних елементів.

Бібліотеку класів реалізовано з використанням технології Windows Presentation Foundation, яка входить до складу .NET Framework 3.0.

Для створення інтерфейсних елементів пульта в роботі пропонується створити доменно-специфічну мову програмування (мову інтерфейсних елементів).

Мова інтерфейсних елементів – мова з синтаксисом та семантикою, яка розробляється для опису всіх можливих компонентів та дій над ними в описуваному домені [11]. Дії описують будь-які процеси, що відбуваються в домені при взаємодії (наприклад, відхилення стрілки індикатору, переміщення важелів та інш.). Доменно-специфічна мова в даному контексті – засіб для опису інтерфейсних елементів з використанням мови XAML (eXtensible Application Markup Language) та бібліотеки класів. Дана мова може бути використана як повторно використане рішення при створенні нових інтерфейсів пультів інструкторів авіаційних тренажерів. Приклад використання доменно-специфічної мови для створення індикатору «тахометр» наведено на рис.8.

```
<ctrls:CircularGuage Radius="100" >
 <ctrls:CircularGuage.Scales>
 <ctrls:CircularScale Radius="99" StartAngle="-225" SweepAngle="55" >
    <ctrls:CircularScale.TickSets>
       <ctrls:CircularTickSet Minimum="0" Maximum="5.25" >
         <ctrls:CircularTickSet.Ticks>
        <ctrls:CircularTickMarkMajor ScalePlacement="Inside"          TickMarkType="Rectangle" />
            <ctrls:CircularTickLabelMajor       StartValue="0"    EndValue="5.2"    ValueSign="Positive"
             Interval="1" ScalePlacement="Inside" />
              </ctrls:CircularTickSet.Ticks>
               <ctrls:CircularTickSet.Pointers>
          <ctrls:CircularPointerNeedle Background="Green" Value="{Binding ElementName=UcElement,
          Path=Value}  />
                </ctrls:CircularTickSet.Pointers>
       </ctrls:CircularTickSet>
     </ctrls:CircularScale.TickSets>
  </ctrls:CircularScale>
 </ctrls:CircularGuage.Scales>
</ctrls:CircularGuage>
```

Рис. 8. Створення приладу «тахометр»

Висновок

Розробка повторно використовуваних рішень – є дієвим способом зниження витрат на реалізацію типових проектів та покращення їх якості. Стан галузі авіаційних тренажерів дозволяє створювати високорівневі, багатократно використовувані рішення програмного забезпечення. Застосування доменного аналізу під час створення програмного забезпечення пульта інструктора авіаційного тренажеру L-410 в Національному авіаційному університеті дає можливість здійснити класифікацію інтерфейсних елементів, побудувати ієрархію абстрактних класів інтерфейсних елементів та розробити доменно-специфічну мову програмування. Рішення-результати, отримані під час проведення доменного аналізу можна багатократно використовувати при побудові інтерфейсів пультів інструкторів різного типу авіаційних тренажерів.

Список літератури

1. Серегин Г.Н. Авиационные тренажеры — реальный путь к повышению безопасности полётов // «Право и безопасность». 2006 № 3-4. С. 20-21)

2. 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.

3. Design of a flight simulator software architecture. Göran Ancker, Jan Wallenberg. – School of

Mathematics and Systems Engineering, Växjö University. – 2002. – 91 p.

4. Желонкин В.И. Система поддержки исследований по выбору и оптимизации видов электронной индикации // Вестник Международной Академии проблем человека в авиации и космонавтике. – М.– 2007. №3(26). – с. 33-39.

5. Сидоров Е.Н. Архитектура программного обеспечения пульта инструктора авиационного тренажера // Матеріали міжнародної науково-практичної конф. аспірантів і студентів «Інженерія програмного забезпечення 2008». – К. – 2009. – с. 45-49.

6. Сидоров Н.А. Повторное использование программного обеспечения // Кибернетика. – 1989. - №3 – с.46-51.

7. Biggerstaff T. Foreword// IEEE Trans, on Software Engineer.v.10, N.5, 1984, p.474-476.

8. Choi S.C., Scacchi W. Extracting and Restructuring the Design of Large System // IEEE Software, Jan., 1990, p.66-71.

9. Torii K., Matsumoto K. Ginder2: An Environment for Computer Aided Empirical Software Engineering // IEEE Transaction on Software Engineering. vol.25, N4, July/august, 1999. – p.474-476.

10. Neighbors J.M. The Draco Approach to Constructing Software from Reusable Components // IEEE Trans. on Softw. Eng. – 1984. – 10, №3. P. 564-576.

11. Prietto-Diaz R. Domain Analysis: An Introduction.- Software engineering Notes. – 1990. – V. 15, № 2. – p. 47–54.

12. Prietto-Diaz R. Domain Analysis For Reusability // COMPASC'87. – Tokyo, Japan. – 1987, Oct. 7-9. p. 23–29.

13. Wartik S., Prieto-Diaz R. Criteria for comparing reuse-oriented domain analysis approaches. – Software productivity consortium, 1991. p.31-67.

14. Сидоров Н.А. Восстановление, повторное использование и переработка программного обеспечения. I. // УСим – К. – 1998. – № 3. – С.74-83.

15. Конституція України.

16. Повітряний кодекс України.

17. Документи ІСАО. Керівництво по аеропортовим службам. Керівництво по запобіганню авіаційних подій. Керівництво по плануванню обслуговування повітряного руху.

18. Авіаційний тренажер ТЛ-410. Технічний опис. Книга 1.

19. Керівництво по льотній експлуатації.

20. Сидоров Н.А., Недоводеев В.Т., Хоменко В.А., Сердюк И.П., Сидоров Е.Н. Реинженерия программного обеспечения информационно-моделирующих тренажерных комплексов. // Управляющие системы и машины. – К. –2008. – № 4. – С.68-74.

21. Меерович Г.Ш., Годунов А.И., Ермолов О.К. Авиационные тренажеры и безопасность полетов. – М.: Воздушный транспорт. 1990. – 343 с.

22. Луцький М.Г., Рябокінь Ю.М. Метод створення програмного забезпечення пульта інструктора АТ. // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук пр. – К.: Век+, – 2009. № 51. С. 78–83.

23. Хоменко В.А., Авраменко Е.А., Рябоконь Ю.Н. Подход к созданию визуального интерфейса пульта инструктора авиационного тренажера. // Матеріали науково-практичної конф. «Актуальні проблеми розвитку авіаційної техніки». – К. – 2009. – с. 117.

24. Луцький М.Г., Сидоров М.О., Рябокінь Ю.М. Реінженерія – шлях підтримки придатності та продовження експлуатації програмного забезпечення авіаційної техніки. //Проблеми програмування. – К. – 2010. №2–3. – с. 229–237.

25. Буч Г. Ообъектно-ориентированный анализ и проектирование с примерами приложений на С++, 2-е изд./Пер. с англ. – М.: «Издательство Бином», СПб.: «Невский диалект», 2001. – 560 с.

26. Ларман К. Применение UML и шаблонов проектирования. – М.:«Вильямс», 2002. –624 с.

Відомості про автора

Рябокінь Юлія Миколаївна - старший викладач кафедри інженерії програмного забезпечення факультету комп'ютерних наук Національного авіаційного університету.

В 2005 році закінчила Національний авіаційний університет за спеціальністю програмне забезпечення автоматизованих систем та отримала кваліфікацію науковий співробітник.

Наукові дослідженння: технології створення програмного забезпечення, шаблонне проектування, доменний аналіз.

Тел. 406-76-41

E-mail: yulia.ryabokin@livenau.net
yulia_r@rambler.ru

Стаття надійшла до редакції 22.05.2010

