Приложна обвивка за университетска информационна система

Мирослав Михайлов, Цветелин Павлов, Венцислав Йорданов

University information system application shell: *This paper revue the application shell, designed for a university information system.*

Keywords: *university information system, application shell, ribbon, fluent user interface, modular application architecture.*

ВЪВЕДЕНИЕ

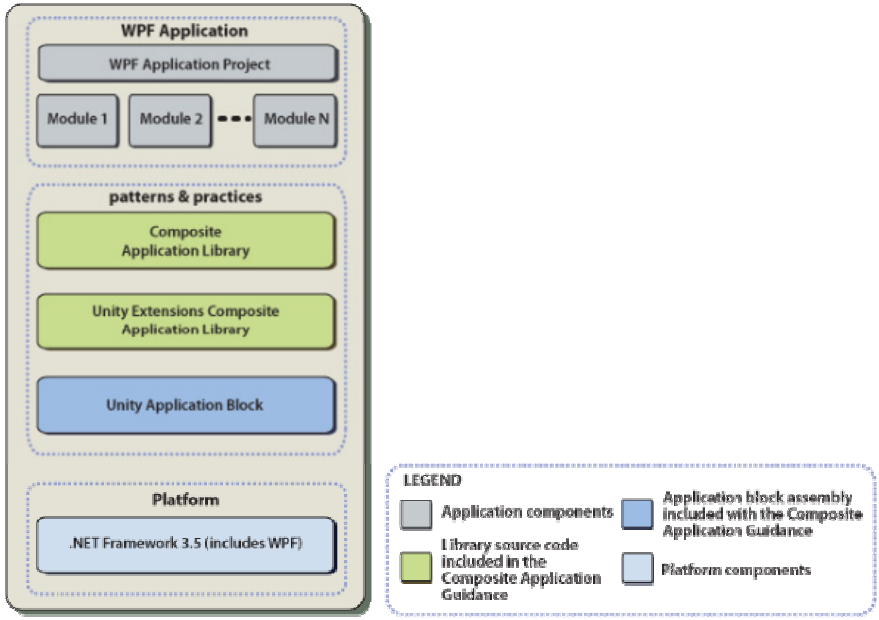
За да отговори на нуждите на университета, университетската информационна система трябва да покрива множество области, да използва данни и да интегрира работни процеси от множество други системи и подсистеми, както и да осигури достъпност на информацията, като изнесе функционалностите си на множество платформи. Докато първите проблеми се решават чрез проектиране на подходяща системна архитектура, последният проблем изисква друг подход. За нуждите на университетската информационна система на Русенския университет беше разработена приложна обвивка, върху която да се базират всички десктоп приложения на системата. Обвивката е реализирана върху модулна приложна архитектура и позволява слабо обвързване между модулите за свързаност, модулите с работна функционалност и модулите на потребителския интерфейс.

Модулна архитектура

Заради разнообразието от модули, които могат да бъдат включени във всяко приложение и необходимостта от слабо обвързване между тях е почти задължително приложната обвивка да бъде базирана на модулна архитектура, т.е. да бъде изградена като композитно приложение.

Композитните приложения се дефинират като приложения, изградени чрез комбиниране на съществуващи функционалности в ново приложение (1). В повечето платформи и рамки за разработка на композитни приложения това се постига чрез разделяне на приложните функционалности в модули, които не са обвързани помежду си, което пък налага използване на „инжекция на зависимости“ (2), композитни събития („наблюдател“) и други известни шаблони за дизайн и микроархитектури. Всеки от модулите обикновено дефинира клас, имплементиращ известен на рамката интерфейс. Този клас бива инстанциран, когато съответния модул бъде зареден и обикновено носи отговорността за инициализацията на модула.

Приложната обвивка на университетската информационна система е изградена върху библиотеките Prism (позната още като Composite Application Block) (3) и Enterprise Library (4). Втората библиотека включва имплементация на „инжекция на зависимости“, а първата дефинира класовете, необходими за изграждане на модули и региони, за вграждане на съдържание в региони и за изпращане на съобщения (събития) между класове в различни модули.



Фиг. 1 Типична модулна архитектура на композитно приложение (5)

Основния подход, следван при проектирането и реализацията на обвивката е обособяване на всяка включена функционалност в отделни модули, като основното приложение (изпълнимия файл) включва само логиката по стартирането, зареждането и инициализацията на модулите, приложната инфраструктура (контейнер за инжекция на зависимости, агрегатор за композитни събития и др.). Така приложението няма знание за библиотеките използвани за изграждане на потребителския интерфейс (т.е. основното приложение „не знае как ще изглежда“), като това се решава от модулите, които се зареждат при стартиране. Това позволява използване на приложната обвивка както за нормално десктоп приложение, така и за изпълнима приложна услуга или за конзолно приложение.

Инициализацията на приложението и модулите е разделена на две нива – първото се осигурява от рамката (Prism) и при нея повечето модули само регистрират класовете си и се абонират за композитни събития. Второто е изпълнено като композитно събитие и осъществява същинската инициализация на модулите – създаване на инстанции, регистриране на гледки в регионите, извикване на уеб услуги и т.н. Второто ниво се извиква единствено и само след като всички модули са инициализирани, т.е. след като всички класове в модулите са регистрирани. По този начин се решават проблемите с зависимостите между модулите – когато същинската инициализация стартира, всички класове, които пораждат зависимости трябва да са вече регистрирани в контейнера за инжекция на зависимости.

Всички модули са силно свързани само към една библиотека от приложния проект – т.нар. „договори“ (“contracts”, “infrastructure” или “core”). Тя съдържа дефиницията на интерфейсите, които могат да породят зависимости. Ако модул използва външна библиотека, то само той има твърда зависимости от нея и външната библиотека трябва да се разпространява с модула, а не с приложението

(такъв е случаят с модулите за потребителския интерфейс). В общодостъпния проект са дефинирани и класове за описание на потребителския интерфейс, които добавят допълнителен слой на абстракция и осигуряват слабо обвързване между имплементацията на приложния интерфейс и модулите, които го използват.

Използваната приложна архитектура и подход позволяват улеснена миграция и адаптация към други среди и платформи. Например, ако на по-късен етап е необходимо да се разработи приложение за информационни терминали[7], то ще постави допълнителни (утежнени) изисквания към потребителския си интерфейс, за да направи работата чрез докосване по-лесна. При нормална реализация, цялото приложение трябва да бъде преработено (може би без класовете за свързаност), докато при така проектираната модулна приложна архитектура е необходимо само да се подмени модулът за потребителски интерфейс. Останалите модули (например модул „Данни за студент“) не е необходимо да бъдат променяни. Ситуацията е сходна и при необходимост от промяна или при корекция на грешка, появила се на по-късен етап.

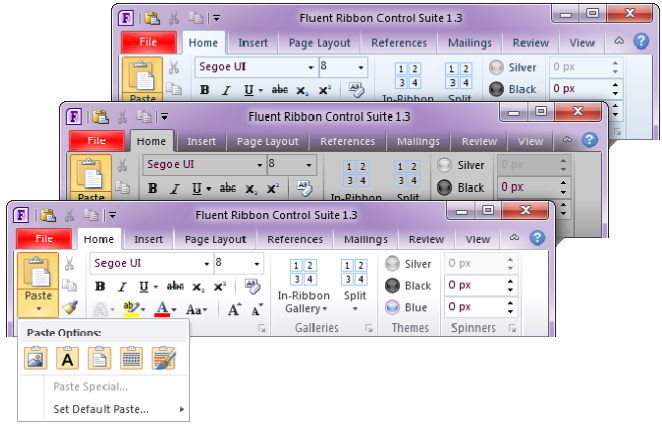
За всички десктоп приложения от информационната система е използван т.нар. „fluent“ интерфейс, както и потребителски интерфейс с прикачващи се прозорци. Двата интерфейса изграждат приложна среда, много сходна с добре познатите приложения от офис пакета на Microsoft, като така се гарантира къса крива на обучение за персонала, който ще използва приложенията на системата.

Потребителски интерфейс тип „Fluent”

Основният принцип на организация на контролите за управление, избран за приложенията на информационната система, е базиран на потребителски интерфейс тип „fluent”, познат още като “ribbon” интерфейс. Основната концепция на този тип интерфейс е обособяване на единственото място за търсене на функционалностите на продукта. При един ретроспективен поглед върху интерфейсите преди “fluent”, лесно се вижда, че за да се намери дадена команда понякога трябва да се прегледат 3-4 нива йерархични менюта и/или да се отворят 3-4 различни ленти с инструменти. Като цяло, досега беше трудно да се формулира стратегия за изграждане на потребителския интерфейс, тъй като нямаше ясно изразена логика в подредбата. „Fluent” интерфейса групира функциите на приложението в логически свързани групи (наричани „табове“, които от своя страна се състоят от „групи“) и така улеснява значително потребителя в намирането на функцията, която му е необходима. „Fluent” интерфейса улеснява потребителите да открият, разберат и ефективно и директно да използват командите с минимален брой натискания на мишката и с време за усвояване, сведено до минимум.

Microsoft официално внедри „fluent” интерфейса в Office 2007 и Office 2010. След това все повече производители базират своите продукти на този тип интерфейс (например AutoCAD 2009 и 2011).

[7] Което по план трябва да бъде разработено.



Фиг. 2 "Fluent" интерфейс (6)

„Fluent" интерфейса се състои от логически групи („табове"), които съдържат отделни панели („групи"). Те съдържат конкретните команди. Всяка програма има различни по вид и съдържание „табове", които се определят от функциите, които тази програма изпълнява. Освен стандартните табове, „Fluent" интерфейса включва и т.нар. „контекстни табове", които са видими само, когато определено съдържание е селектирано. Например, ако от група потребители на екрана е избран студент, то може да се появи контекстен таб „Студент", който да съдържа специфична команди само за студенти. Като цяло структурата на „Fluent" интерфейса групира предишното използване на менюта и панели с бутони, познати от интерфейсите с ленти с инструменти във функционални единици, които позволяват търсената от потребителя команда да бъде намерена с минимален брой цъкания на мишката и с един преглед на видимите панели.

Използването на „Fluent" интерфейса води до взаимодействие, близко до взаимодействието с продуктите от офис пакета на Microsoft. Същевременно, възможността за моделиране на контекста от дава необходимата свобода за изграждане на функционалностите на приложенията.

В общия случай, когато потребители използват софтуер за първи път, изпитват затруднения с командите и възможностите му. Целта на „fluent" интерфейса е да сведе до минимум времето за изучаване, изхождайки от факта, че повечето потребители, които ще работят със системата, вече са запознати с този тип интерфейс като елементи, структура и принцип на действие.

Отделните модули имат специфични команди, отнасящи се само за функционалностите в тях. Централизирането на всички команди на едно място, без значение от заредения модул, улеснява потребителя в извършване на желаните от него действия и дава свобода на изграждане на потребителския интерфейс, независимо от логическата структура на приложението и модулите. Това също така ни гарантира, че потребителите ще имат достъп до цялата функционалност на заредения модул в едно единствено навигационно поле, дори ако дизайнът на самия модул се промени.

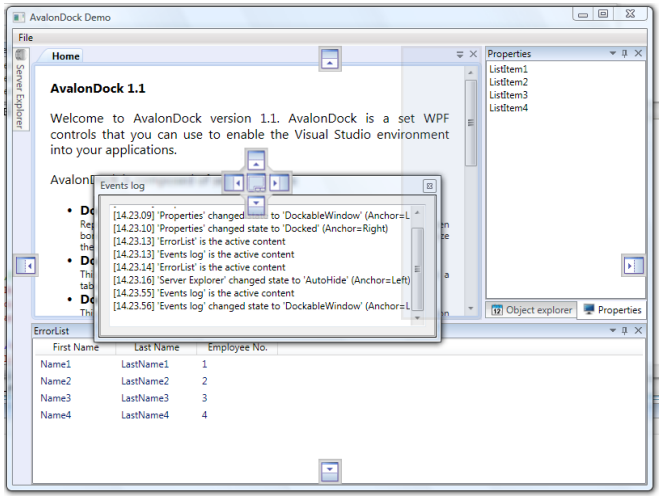
Броят на командите също е фактор. Ако някой определен модул има повече функции от друг, възможността за промяна на големината на иконите и за разделяне на функциите в допълнителни групи и контекстни табове помага да се визуализират

всичките функции без да се налага да се отварят допълнителни менюта или ленти с инструменти.

В приложната обвивка е използвана библиотеката за “fluent” интерфейс Fluent Ribbon Control Suite (6). Специфичните за библиотеката функционалности са обособени в отделен модул, което позволява всички останали модули на приложната обвивка да бъдат изградени без да се взимат предвид особеностите на тази конкретна библиотека.

Прикачващи се Пана

Докато “fluent” интерфейса се грижи за правилна и предсказуема подредба на потребителските команди, приложената обвивка има необходимост от подредба и на съдържанието, която ще осигури гъвкав интерфейс, способен да се нагласи според нуждите на потребителя.



Фиг. 3 Интерфейс с прикачващи се пана (7)

За управление на съдържанието в приложната обвивка се използва интерфейс с прикачващи се пана (прозорци). Това позволява информацията да бъде визуализирана логически според съдържанието си и смисъла, който носи. Чрез функциите за контекстни табове на „fluent” интерфейса се намалява значително броя на първоначално визуализираните команди, като те се появяват „само когато ти трябват“.

В приложната обвивка е използвана библиотеката за прикачващи се пана AvalonDock (7). Специфичните функционалности и класове на библиотеката са обособени в отделен модул, като използването им става през абстрактния слой с класове по същия начин както и при компонентите за навигация (командите).

ЗАКЛЮЧЕНИЕ

Използването на познат и актуален потребителски интерфейс позволява на потребителите бързо да усвоят продукта, с който ще работят. Това, съчетано с правилно проектирана приложна архитектура позволява улеснено развитие, последваща разработка и отстраняване на грешки – както софтуерни, така и причинени от потребителите. Директния ефект за университета се изразява в повишено бързодействие на потребителите и повишено качество на обслужването.

ЛИТЕРАТУРА

[1] Composite application. *Wikipedia.* [Онлайн] Wikimedia Foundation, Inc.,, 28 март 2010 г. [Цитирано на: 27 октомври 2010 г.] http://en.wikipedia.org/wiki/Composite_application.

[2] Dependency injection. *Wikipedia.* [Онлайн] Wikimedia Foundation, Inc.,, 19 октомври 2010 г. [Цитирано на: 26 октомври 2010 г.] http://en.wikipedia.org/wiki/Dependency_injection.

[3] patterns & practices: Prism. *CodePlex.* [Онлайн] Microsoft Corporation, 2010 г. http://compositewpf.codeplex.com/.

[4] Microsoft Enterprise Library. *MSDN Library.* [Онлайн] Microsoft Corporation, юли 2010 г. http://msdn.microsoft.com/en-us/library/ff648951.aspx.

[5] Composite Application Library. *MSDN Library.* [Онлайн] Microsoft Corporation, 2010 г. http://msdn.microsoft.com/en-us/library/ff647752.aspx.

[6] Fluent Ribbon Control Suite. *CodePlex.* [Онлайн] 2010 г. http://fluent.codeplex.com/.

[7] AvalonDock. *CodePlex.* [Онлайн] Microsoft Corporation, 2010 г. http://avalondock.codeplex.com/.

[8] Patterns For Building Composite Applications With WPF. *MSDN Magazine.* [Онлайн] Microsoft Corporation, септември 2008 г. [Цитирано на: 27 октомври 2010 г.] http://msdn.microsoft.com/en-us/magazine/cc785479.aspx.

За контакти:

доц. д-р Мирослав Михайлов, директор на ЦИКО, Русенски университет „Ангел Кънчев“, тел.: 082 888 782, e-mail: mmihaylov@uni-ruse.bg.

маг. инж. Цветелин Павлов, ЦИКО, Русенски университет „Ангел Кънчев“, тел.: 082 888 328, e-mail: cpavlov@uni-ruse.bg.

маг. инж. Венцислав Йорданов, ЦИКО, Русенски университет „Ангел Кънчев“, тел.: 082 888 328, e-mail: vkiordanov@uni-ruse.bg.

Публикацията е реализирана по проект по ФНИ 2010 РУ-02.

Докладът е рецензиран.