Структурно и обектноориентирано проектиране на информационни системи

1. Структурно проектиране

Структурно проектиране (structured design) свързано с определяне на структурата на информационните системи (ИС). Тя се представя като йерархична структура от функционално обособени модули (подпрограми, процедури, функции). Модулите обикновено са реализация на функционална абстракция. Всяка абстракция отразява определена функционалност на ИС, която се представя чрез абстрактна процедура, функция или операция. Модулите са свързани с едно или краен брой действия, които преобразуват входните данни в изходни.

При структурното проектиране се използват структурни диаграми и спецификация на модулите. Като нотации се прилагат функционални дървета и диаграми на потока от данни (DFDs-Data Flow Diagrams).

Структурните диаграми представят графично модулите на системата и взаимодействието между тях. Визуално функционалните модули се представят с правоъгълници. Входните и изходните параметри на модулите и връзките между тях се представят със стрелки. Така се представят потока от данни и структурата на извикване (фиг.1).

Фиг. 1 Структурна схема на информационна система

2. Обектноориентирано проектиране

Обектноориентираното проектиране (object oriented design) може да се разглежда като продължение на обектноориентирания анализ. То използва същите нотации (методи за описание) като анализа. Основно при този вид проектиране се използват диаграми на класовете. Могат да се използват и диаграми на действия и крайни автомати.

Обектноориентираното проектиране е свързано с проектиране на архитектурата и на разработката на информационните системи. Първоначално се определя архитектурата на системата. Тя зависи от предназначението на системата, потребителите и потребителския интерфейс, начина на управление на данните, начина на съхранение и обработка. Например, ако данните се съхраняват в обектноориентирана база от данни, свързването на модела към потребителския интерфейс зависи от избраната система за управление на база от данни (СУБД).

За разработка на архитектурата на ИС основно се използват обектноориентираният модел, създаден при дефиниране на системата и интерфейсите към околната среда.

Връзката към потребителския интерфейс се реализира на ниво класове или на ниво обекти. Класовете на потребителския интерфейс се приемат като класове на приложението. Те трябва да бъдат интегрирани към съществуващата вече йерархия на класовете на приложението. В графичното изобразяване на архитектурата на системата за отделяне на класовете на потребителския интерфейс от класовете на приложението се използва прекъсната линия.

При проектиране на разработката се детайлизира предложената архитектура и се прави настройка към правилата на избрания език за програмиране. Обикновено кода на приложението се генерира автоматично от диаграмата на класовете чрез използване на подходяща за целта програмна среда.

Когато ИС се проектира за работа в компютърна мрежа обектноориентираният модел може да се декомпозира на два отделни модела, клиентски и сървърен. Клиентският проектен модел се разширява с класовете на потребителския интерфейс.

3. Проектиране на ИС на базата на компоненти

Проектирането на ИС на базата на компоненти се основава на интегрирането на вече съществуващи софтуерни компоненти, както и на разработването на софтуерни компоненти, които могат да бъдат повторно използвани в различни приложения.

Основната цел на компонентно базираното проектиране е да осигури методи и средства за: изграждане на софтуерните системи от компоненти; изграждане на компонентите като единици, които могат да се използват повторно; поддържане на системите чрез замяна на компонент и/или добавяне на нов компонент.

Софтуерните системи се разглеждат като системи в които са интегрирани или композирани слабо свързани независими компоненти. Основните елементи на тези системи са:

- ✓ независими компоненти, специфицирани от техните интерфейси;
- ✓ стандарти за компоненти, които определят правилата за интеграция на компонентите;
- ✓ среда за оперативна съвместимост (inter operability) на компонентите.

Според изследователите под компонент се разбира малка единица, която има ясно определени функционални и нефункционални характеристики. Представителите на индустрията считат, че компонент е немалък софтуер, които може да се използва повторно, има структура и интерфейси, специфични за приложната област.

През 2002г. Сиперски (Szyperrski) предлага следната дефиниция за компонент: " Софтуерен компонент е единица за композиране с интерфейси, които се определят на базата на договори и зависи от контекста. Софтуерният компонент може да се внедрява самостоятелно и може да бъде обект на композиция от трети страни."

От това определение следва, че софтуерният компонент градивна единица на софтуерните системи. Той се дефинира чрез своите интерфейси, които обикновено са два вида: интерфейс осигурявам (provides interface) и интерфейс извиквам (requires interface). Тези интерфейси се определят от гледна точка на взаимодействието между компонентите и от ролята на самите компоненти.

Спецификацията на средата за разработка и внедряване, както и работната среда определят контекстните зависимости за компонентите и техните интерфейси.

Според Хейман и Коунсил (Heiman, Council, 2001) софтуерният компонент е "софтуерен елемент, който съответства на компонентен модел и може да бъде независимо внедрен и композиран без промени, съгласно стандарт за композиране". Компонентният модел е дефиниция за стандарти за разработване, документиране и внедряване на компоненти. Някои от известните и разпространени компонентни модели са:

- ✓ EJB model (Enterprise Java Beans) на фирмата Sun;
- ✓ COM+ model (.NET model) на фирмата Microsoft;
- ✓ Corba Component Model на организацията OMG;
- ✓ Koala на фирмата Philips.

Като основни характеристики на софтуерните компоненти могат да се посочат:

- независимост - трябва да може да се внедрява независимо от другите компоненти;
- стандартизиран - трябва да се съответства на избран компютърен модел за да е част от процеса на разработка на софтуера на системата;
- документиран - ясно и изчерпателно, максимално удобно за потребителите;
- готовност за внедряване - да не се налага да се компилира, а директно да се внедри в системата или да работи самостоятелно;
- готовност за композиция - готовност за участие в композиция с други компоненти, да има публично дефинирани интерфейси.

Елементите на компонентния модел могат да бъдат разделени на следните групи:

- ✓ интерфейси елементи - определят, как трябва да се дефинират интерфейсите на компонентите;
- ✓ за използване - елементи, които съдържат необходимата информация за приложение на компонентите. Те определят правилата за именуване на компонентите, атрибутите на компонентите и др.
- ✓ за внедряване - показват правилата за използване, документиране, внедряване, за замяна на един компонент с друг и др.

Съществена характеристика на компонентно базираното проектиране е независимост на процеса на разработка на компонентите от процеса на разработка на софтуерните системи. Това дава възможност тези два процеса да протичат паралелно.

Моделът на компонентно базираното разработване на софтуер е еволюционен по своята същност. Това налага използването на итеративни подходи. Основните дейности при разработване на системи базирани на компоненти са: намиране и избор на компонент; тестване; адаптиране; внедряване и паралелно разработване на компонент (фиг.2). При проектиране на софтуерни системи без използване на компоненти основните дейности са: анализ и дефиниране на изискванията, проектиране на системата, кодиране, тестване, внедряване и поддръжка. Общото и различното между тези дейности при двата подхода може да се обобщи, както следва:

Препоръчително е изискванията към системата да са по-общи в началото с което ще се улесни избора на компоненти. Допълнителна дейност при компонентно базираното разработване е идентифициране на компонент. Тя се състои от няколко действия - търсене, избиране и тестване. Тестването е свързано с разработка на тестови случаи, които да проверят дали компонентът работи според очакванията. Ако той покрива само частично тези изисквания е необходимо: компонентът да се адаптира към определена спецификация; спецификацията да се адаптира към компонента; да се разработи нов компонент.

Проектирането на системата при компонентния подход се определя от идентифицирането на компонентите. Архитектурата на системата зависи от спецификацията на система, избора на компоненти и избрания компонентен модел.

Кодирането на системата се свежда до кодиране на слепващ код (glue code) за връзка между компонентите и до тяхното адаптиране.

Интегрирането на системата е композиране на създадените и избраните компоненти в една система. Възможни са различни типове композиране - последователно, йерархично или адитивно. Интегрирането е свързано с: адаптиране на компонентите; реконфигуриране на композициите, при необходимост; промяна на свойствата на системата. Адаптирането може да се направи по различни начини, например чрез: специфициране на определени параметри; разработка на обвивка (wrapper), която да управлява

входовете и изходите на компонента; разработка на нов компонент, които да контролира даден компонент. Реконфигуриране на композициите може да се наложи при възникване на конфликти и несъвместимост между отделни компоненти.

Софтуерните системи, както и ИС, разработени на базата на компоненти се поддържат по-лесно. В процеса на тяхното използване един компонент може да се замени с друг, както и да се добави нов компонент, без да е необходима компилация на системата.

Като предимства компонентно базираното проектиране могат да се посочат:

- по-голяма степен на използваемост;
- по-голяма съвместимост;
- намалено време за разработка на системите;
- компонентите са относително независими;
- промяната на един компонент не налага промяна на останалите;
- компонентите си взаимодействат чрез интерфейси;
- по-лесна поддръжка;
- по-ниска цена и др.

Като недостатъци на компонентно базираното проектиране могат да се посочат:

- необходимо е време и ресурси за разработване на компонент;
- невинаги изискванията към системите са ясни и недвусмислени, което затруднява проектирането и реализацията на компонентите;
- възможно е да има противоречие между използваемост и многократна използваемост на компонентите.