Разработка на софтуер – Доклад

**1.Потребителски интерфейс**

Потребителски интерфейс в индустриалния дизайн е мястото за взаимодействие между човека и машината с цел да се позволи ефективно управление и контрол върху машината от страна на човека, като в същото време машината връща обратно информация, която подпомага процеса на вземане на решения от оператора. Примерите, които описват това широко понятие за потребителски интерфейс, включват: интерактивните аспекти на компютърните операционни системи; инструментите за ръчно управление; контролните табла за управление на тежката механизация в различни отрасли на промишлеността и други механизми за управление на процеси. При създаването на потребителските интерфейси играят роля дисциплини като ергономия и психология.

Целта на проектирането на потребителски интерфейс е да се произведе продукт, който да направи лесна, ефективна и приятна работата с машините, така че да се достигне до желания резултат. Това обикновено означава, че операторът трябва да осигури минимално количество входни данни за постигането му, като в същото време се свежда до минимум обратната връзка от страна на машината.

С по-широкото навлизане на персоналните компютри терминът потребителски интерфейс все по-често се припокрива с графичния потребителски интерфейс (на английски: graphical user interface, GUI) на приложния софтуер. При проектирането на контролни системи и механизация за промишлеността, потребителският интерфейс се нарича човеко-компютърен интерфейс (на английски: human-computer interface, HCI) или човеко-машинен интерфейс (на английски: man-machine interface, MMI).

Човеко-машинният интерфейс управлява взаимодействието между човека и машината. Мембранните превключватели, гумените клавиатури и сензорните екрани са само някои примери за физическите елементи на човеко-машинния интерфейс, които могат да се видят и докоснат.

В по-сложните системи интерфейсът човек-машина обикновено е компютризиран. Терминът човеко-компютърен интерфейс се отнася до такъв вид система. В контекста на информационните технологии значението му обичайно включва и софтуера, осъществяващ контрол върху физическите елементи на взаимодействието човек-компютър.

Проектирането на човеко-машинните интерфейси е свързано и с отчитане на човешкия фактор. Съответните дисциплини са ергономия (на английски: human factors engineering и ползваемост, които са част от системното инженерство. Инструментите, използвани за това, почиват на познанията по информатика, като например компютърна графика, операционни системи, езици за програмиране. В днешно време, тъй като почти всеки човеко-машинен интерфейс използва не команден ред, а графики и изображения, все по-често се използва израза графичен потребителски интерфейс.

**2.Свързване на данни в графичен интерфейс**

Свързването на данни (Data Binding) осигурява автоматично прехвърляне на данни между контроли и източници на данни. Можем например да свържем масив, съдържащ имена на градове с ComboBox контрола и име­на­та от масива ще се показват в нея.

При добавянето на свързване указваме свойството на контролата, което ще свързваме с данните, източника на данните и път до списък или свой­ство на източника, към което ще се свържем. Този път може да е име на свойство, йерархия от имена разделени с точки или празен низ. Ако пътят е празен низ ще се извика метода ToString() на обекта използван като източник на данни.

Промяна на дадено свързано свойство от дадена контрола, променя дан­ни­­те в източника, към който то е свързано. Обратното не е вярно. При про­мя­на на източника на данни, свързаните към него контроли не си про­ме­нят свойствата.

Ако сме променили данните в източника на данни и искаме да отразим промените в свързаните с него контроли, трябва първо да премахнем (изтрием) свързването и след това да го добавим отново.

**-Видове свързване**

Просто свързване (simple binding) – свързване на контрола с еди­ни­чен обект или единичен (текущ) елемент от списък. Такова свър­зва­не използваме обикновено с контроли като TextBox и CheckBox, кои­то показват единична стойност.

Сложно свързване (complex binding) – свързване на списъчна кон­трола със списък. Такова свързване използваме с контроли като ListBox, ComboBox и DataGrid. При него се поддържа текущо избран (активен) елемент от списъка.

**3.Принципи за правилно разделяне на компонентите на приложението по слоеве**

В софтуерното инженерство многослойната архитектура (наричана още N-слойна архитектура) е архитектура от тип клиент-сървър, в която интерфейсът, обработката на приложения и съхранението и обработката на данни са логически разделени на отделни модули. Най-разпространената форма на многослойна архитектура е трислойната архитектура.Многослойната архитектура осигурява модел, по който разработчиците могат да създават гъвкави приложения, които могат да бъдат използвани многократно. При разделянето на едно приложение на слоеве, разработчиците имат възможността да добавят или променят отделен слой, вместо да преработват цялото приложение.

Всички съвременни бизнес приложения имат достъп до различен вид данни като част от тяхната основна функционалност. След като релационните бази данни добиха широка популярност, IT индустрията се преориентира от еднослоен модел (mainframe) към модела клиент-сървър. При него са налице клиент, който включва презентационна логика и голямата част от бизнес логиката на приложението, както и сървър включващ в себе си хранилище за данни (data storage) и част от бизнес логиката под формата на готови и съхранени заявки към базата данни. От началото на 90-те години клиент-сървър моделът започна да излиза от употреба поради високите разходи за поддръжка и невъзможността за разпределение на отговорности при възникване на проблеми.

Трислойната архитектура обикновено се състои от презентационен слой (потребителски интерфейс) на най-високо ниво в приложението. Той служи за прякото взаимодействие с потребителя и изпращането на заявки към бизнес слоя. Не трябва да има директна връзка между този слой и слоят за бази данни. Бизнес слоят служи за обработка на данните и работните процеси. Той комуникира както с презентационния слой, така и с базите данни. Слоят за данни комуникира само с бизнес логиката и служи за съхранение данни и тяхното използване. Често отделните модули се разработват и на различни платформи. Разработена е от Джон Дж. Донован, в неговата компания OEC (Open Environment Corporation), основана в Кеймбридж, Масачузетс.

Освен стандартните предимства на модулния софтуер, трислойната архитектура е проектирана да позволява да бъде променен или заменен независимо, който и да е от трите слоя, за да отговаря на промени в изискванията или технологиите. Например, промяна на операционната система на презентационния слой би засегнала само кода за потребителският интерфейс.

В повечето случаи, потребителският интерфейс работи на десктоп система и използва стандартен графичен интерфейс, функционалната логика за обработка на данните, която може да се състои от един или повече отделни модула, работещи на десктоп система или сървърно приложение, и РСУБД (релационните системи за управление на бази данни) работещи на сървър база данни или мейнфрейм, който съдържа логиката за съхранението и обработката на базите данни. Междинният слой може самият да бъде многослоен. В такъв случай цялостната архитектура се нарича N-слойна архитектура.

Обмяната на информация между слоевете е част от архитектурата. За целта може да се използват един или повече от следните протоколи – SNMP, CORBA, Java RMI, .NET Remoting, Windows Communication Foundation, sockets, UDP, уеб приложения или други стандартни или подходящи протоколи. Често за свързване на отделните слоеве се използва помощен софтуер, наричан middleware. Различните слоеве често (но не винаги) работят на различни сървъри, и всеки слой може да работи на система от свързани компютри, функциониращи като едно цяло, наричани cluster.

**4.Изграждане на интерфейси към приложение**

Можете да създадете и публикувате интерфейс на приложение, който определя връзките, които другите собственици на приложения трябва да използват, за да се свържат с вашето приложение. Интерфейсът на приложението включва сървърите и услугите, необходими за свързване с вашето приложение. Той също така показва на други собственици на приложения къде трябва да добавят своите сървъри, за да изградят необходимите връзки към вашето приложение.

За да използвате интерфейси на приложения, трябва:

-Създайте интерфейса на приложението :

-Създайте интерфейс на приложението с име и незадължителен коментар.

-Добавете интерфейсни връзки към интерфейса на приложението, които включват сървърите и услугите, необходими за достъп до приложението.

-Публикувайте интерфейса на приложението, за да го направите достъпен за други приложения.

-В друго приложение създайте връзката към интерфейса на приложението :

-Други собственици на приложения създават връзки към приложението в своите приложения и избират необходимия интерфейс на приложението.

-Собствениците на приложения добавят своите сървъри към връзката към приложението и отварят билет, за да разрешат достъпа.

Например всички уебсайтове във вашата организация трябва да се свържат с база данни. Приложението за база данни изисква уеб сървърът да се свърже със сървъра на базата данни и със сървър за удостоверяване на потребителя. Като собственик на приложение за база данни можете да изградите интерфейс, който включва връзка към сървъра на базата данни с правилните услуги и друга връзка към сървъра за удостоверяване на потребителя. Сега всеки собственик на уебсайт във вашата организация може да използва създадения от вас интерфейс, за да добави необходимите връзки, а собственикът на уебсайта просто трябва да добави уеб сървърите, които се използват в уебсайта.