Komplex rendszerek készítése

írta: Sólyom-Nagy Péter

Bevezetés

Szakmai munkám során sokféle módszertant kipróbáltam már, mindenből tanultam egy kicsit, a praktikus dolgokat átemeltem más környezetbe is. Ezek alapján összeállítottam egy rövid vázlatot arról, hogy hogyan érdemes egy komplex rendszer tervezésekor az alapokat elkészíteni.

Nincs tökéletes módszertan, ami minden fejlesztésnél egységesen meghozza a várt eredményt, ezért ennek az írásnak nem is célja az „aranycsinálás” definiálása. Ez a cikk egy a sok közül, ami támpontot nyújt azoknak, akik újak ezen a téren, vagy bővíteni szeretnék tudásukat.

Én ha elakadok, az internetet kezdem el böngészni segítségért. Ilyenkor jól jön, ha találok egy fórumot, vagy egy olyan írást, ami kifejezetten a problémámmal foglalkozik.

Mivel a munkáim során elsősorban adatbázis alapú, többfelhasználós rendszerekkel dolgoztam, ez az írás is ezt a vonalat fogja követni.

A feladat megfogalmazása

Az első legfontosabb feladat, hogy tisztán lássuk a rendszerrel szemben támasztott elvárásokat. Egy kis alkalmazásnál, amit magunknak készítünk talán ez a lépés nem olyan fontos, de egy összetettebb rendszernél ez már elengedhetetlen. Ez különösen igaz, ha a projekten többen dolgoznak.

A feladat megfogalmazásakor a felhasználó a fontos. Ilyenkor még azt kell figyelni, hogy a felhasználónak mire van szüksége, és milyen úton szeretné a végeredményt elérni. Ez a fázis leginkább egy beszélgetés a felhasználóval. Ilyenkor hallgatni szoktam a felhasználó beszámolóját, hogy mit szeretne.

A legtöbb esetben egy új szoftver fejlesztésére akkor kerül sor, amikor az adatok már nem férnek el egy Excel táblában. Ez abból a szempontból előnyös, hogy nem a nulláról indulunk, hanem van egy konkrét elképzelés, illetve a felhasználó meg tudja fogalmazni a teljes folyamatot. Ez az ideális eset.

Persze nem mindig vagyunk ilyen szerencsések. Néha nagyon nehéz kitalálni, hogy mi is a feladat. Sok múlik a konzulens fogalmazási készségén.

Nekünk az a feladatunk, hogy kihámozzuk a lényeges pontokat, és közben megértsük a teljes folyamatot. Ha valami nem világos, fontos, hogy a lehető leghamarább tisztázzuk, mert teljesen rossz irányba viheti el a teljes kivitelezést. Utólag sokkal nehezebb javítani.

Tervezés

Ha ismert a feladat, nekiláthatunk a tervezésnek. Tervet azért kell készíteni, hogy mindig legyen egy kapaszkodó, ami vezet a fejlesztés során. Ha nincs terv, az embernek menet közben sokminden eszébe jut és lehet, hogy másik irányba indul el, mint az eredeti cél. A tervvel ezt valamennyire kordában lehet tartani. Természetesen az új ötletek között hasznosak is lehetnek, de ilyenkor vissza kell térni a tervezéshez, és úgy kell beilleszteni az újítást, hogy a rendszer integritása minél kevésbé sérüljön.

Sokféle tervezési módszer van, ezekről sok írás található az interneten. A véleményem az, hogy mindenkinek ki kell alakítani a saját módszerét. A lényeg, hogy működjön.

Tervezés adatbázissal

Az én módszerem, hogy a feladat ismeretében elkezdem tervezni az adatbázist. Ezzel lehet vitatkozni, de a gyakorlatban bevált. Persze ehhez szükséges némi rutin abban, hogy egy-egy objektum a felhasználói paramétereken kívül milyen technikai paraméterekkel kell, hogy rendelkezzen. Ilyen például az egyedi azonosító, annak típusa, az objektumok közötti reláció jellege, vagy a szabályok érvényessége.

> ***Néhány ökölszabály***
>
> - Minden táblához javasolt egy egyedi azonosító beillesztése
> - Az egyedi azonosító általában egy folyamatosan növekvő numerikus érték, de ha egy szabálytábláról van szó, ami csak pár tételt tartalmaz, akkor az azonosító lehet szöveges. Ez megkönnyíti a karbantartást.
> - Ha definiálnunk kell szabályokat, amiknek a paraméterei időnként változhatnak, hozzunk létre egy főtáblát, amiben a szabályok vannak, és egy altáblát, amiben a változó paraméterek. Így a visszamenőleges lekérdezések is konzisztensek maradnak.
> - A szabályok altábláiban a kulcs két részből áll, a főtábla egyedi azonosítójából és egy érvényesség kezdete dátumból. Ezzel a módszerrel mindig lesz értéke a szabálynak és mindig csak egy.
> - Ha öszerendeléseket kell definiálnunk két objektumhoz, a két objektum táblája közé hozzunk létre egy kapcsolótáblát. A kapcsolótáblában szerepelnie kell a két főtábla kulcsának, valamint a tól és ig dátumoknak.
>
> A sor folytatható még mindenkinek ízlése szerint.

Azzal, hogy elkezdem létrehozni a táblákat, egyben kialakul egy kép a rendszer működéséről is.

Fejlesztés

A terv birtokában neki lehet kezdeni a fejlesztésnek. A fejlesztés alatt íródik a kód, itt alakul ki maga a rendszer. Ha jó tervet készítettünk, a fejlesztés már csak egy forgatókönyvet hajt végre. A valóság persze nem mindig így van…

> ### A lépések sorrendisége
>
> A tapasztalatok azt mutatják, hogy a gyakrolatban a tervezés közben már elég sok kód íródhat, ami nem baj de figyelni kell arra, hogy ez ne korlátozzon. Ha menet közben kiderül, hogy a tervezett út nem járható, merjünk visszanyúlni az alapokig és teljesen újragondolni a tervet, ne akadályozzon, ha már kidolgoztunk néhány metódust.
> Az alapos tervezés csökkenti a kompromisszumos megoldásokat, ami hatékonnyá teszi a rendszer működését.

Konvenciók

Mindenkinek megvan a saját fejlesztési stílusa. Ez értendő a kód elrendezésére, a logikai felépítésre, de a rendszer egészére is. A konvenciók abban segítenek, hogy egységes, átlátható legyen a forráskód és a rendszer működése.

Teljesen mindegy, hogy milyen alapszabályokat határozunk meg, de különösen csoportos fejlesztésnél, a szabályokat igyekezzünk mindig betartani. Konvenciók nélkül a kód utólag sokkal

nehezebben értelmezhető, mert újból „meg kell tanulnunk”, hogy eredetileg mi szerint neveztük el a változókat.

> *A konvenciók használata a programnyelvtől és a szerkesztőprogramtól független, de azt meg kell említenem, hogy a Visual Studio 2008 a C# forráskódokat magas szinten formázza, ezzel nagymértékben megkönnyítve a fejlesztő munkáját.*

A konvenciók a forráskódban vonatkozhatnak a változók elnevezésére, a definíciók, metódusok sorrendjére, vagy bármi egyébre, ami a rendezettséget segíti. Nem szabad azonban túlzásokba esni. A konvenciók a munka segítésére, nem a bonyolítására vannak kitalálva. *Semmi értelme például annak, hogyha megkötjük, hogy a változók neve tartalmazza a létrehozás dátumát is.*

Kommentelés, megjegyzések

Minden programnyelvben van lehetőség megjegyzés beszúrására a kódba. Ezt használjuk is ki.

Szakértők szerint akkor jó egy forráskód kommentelése, ha a kommentek a teljes forráskód 20-30 százalékát teszik ki. Gyakorlati tapasztalat, hogy nem lehet túlkommentezni egy kódot. Komplex metódusokban sokszor az is előfordul, hogy a forráskód minden második sora komment.

A kommentnek lényegre törőnek kell lennie, és nem a nyelvtani helyesség a fontos. Olyan megjegyzéseket kell írni, amik az utólagos olvasásnál vissza tudják adni a programozó gondolatmenetét, így könnyű a programozás folytatása, vagy javítása.

Én a gyakorlatban minden publikus metódus, tulajdonság interfészét kommentelem (Ezt követelei meg a C# fordító), de néha a privát metódusoknál is jól jön ez. A metódusok tartalmát a láthatóságuktól függetlenül kommentelem, itt az a döntő, hogy a metódus hány soros, iletve mennyire triviálisak a parancsok. Ha a metódus működése kiemelten fontos (például egy komplett jelentés, vagy egy automatizált adatfrissítés), akkor jön a soronkénti megjegyzés beszúrása.

Milyen legyen a megjegyzés?

A megjegyzés legyen lényegretörő és informatív. Annak semmi értelme, hogy egy változó létrehozását úgy kommentelem, hogy „`Változó létrehozása`”. Ha viszont azt írom, hogy „`A számla összegek kumulált értéke`”, akkor már rögtön tudom, hogy ebben a változóban a feldolgozott számlák teljes értékét fogom megkapni a feldolgozás végén.

Mikor kell a kommenteket létrehozni?

Amint lehet. Véleményem szerint a megjegyzések létrehozása egy időben a metódus létrehozásával a fejlesztési időt maximum 1-2 százalékkal növeli, míg ha utólag próbáljuk a működést „újból” kitalálni, a megjegyzések létrehozása 10-20 százalékkal emeli a ráfordított időt, ráadásul sokkal felületesebb lesz.

Tesztelés

Az elméleti fejlesztési szabályok szerint a fejlesztést követi a tesztelés, de a gyakorlatban a tesztelés már menet közben kell, hogy elinduljon. Én ha megírok egy metódust, azonnal látni akarom, hogy jó-e, amit írtam. Ha csak a végén tudom kipróbálni, már nem biztos, hogy emlékezni fogok, mit-miért írtam.

Persze egy összetett műveletsort, vagy feldolgozó folyamatot a legvégén is ellenőrizni kell. A tesztelésnek két fő változata van; a fejlesztői teszt és a felhasználói teszt.

A fejlesztői teszt elsősorban a fejlesztő szemszögéből kell, hogy történjen. A fejlesztő azt tudja, hogy a kódja milyen logikát követ, vagyis azt kell kipróbálnia, hogy az egyes elágazások mindegyike helyesen működik-e, a követelményeknek megfelel-e.

A felhasználó már csak akkor fog tesztelni, amikor a rendszer (vagy az adott modulja) összeállt és a felhasználó is tudja már rendeltetésszerűen kezelni. A felhasználó nem ismeri a rendszer logikáját, őt csak az érdekli, hogy az ő elképzelése szerinti eredményt hozza. Ő a napi gyakorlatot fogja követni, vagyis olyan adatokkal fog tesztelni, amik életszerűek.

Meglepő lehet, hogy néha a felhasználó képes olyan hibákat kihozni, amik a fejlesztőnek eszébe sem jutnak.

> **Én nem is ezt akartam…**
> A teszteléskor jöhet elő a fejlesztők rémálma, amikor a felhasználó rájön, hogy ez nem is az, amit szeretett volna. Az okot nehéz megtalálni, lehet, hogy a felmérésnél voltak félreértések, de gyakori az is, hogy a felhasználó sem tudja, mit akar. Ezek a hibák csak felmérési, tervezési rutinnal csökkenthetők, de nem küszöbölhetők ki teljesen.

A komplex tesztelések eredményét célszerű dokumentálni. Ez a papírmunka arra jó, hogy megfogalmazzuk a feltárt hibákat, az új igényeket. Amikor visszatérünk a fejlesztési (ritkább esetben a tervezési) fázishoz, készen van a feladat, amit meg kell oldani.

Dokumentáció

Ahhoz, hogy a rendszer teljes legyen, dokumentálni kell. A dokumentáció készítését sokszor elhanyagolják, pedig nagyon hasznosak.

A dokumentáció részletességét, mennyiségét nem lehet általánosan megkövetelni, ez függ a cégen belüli szokásoktól, illetve az adott projekttől. Ha van egy jól bevált rendszer, nyugodtan kövessük!

A fejlesztés elején célszerű egy rövid dokumentumot írni arról, hogy milyen indíttatásból kezdünk bele egy fejlesztésbe. Ebben le kell írni, hogy kinek fejlesztünk, és mi a fő célja a rendszernek. A részletek itt nem fontosak, csak a fő irány.

A részleteket egy újabb dokumentumban kell összefoglalni. Ahhoz, hogy a feladat adott legyen, előbb fel kell mérni a jelenlegi állapotot. Nagyon leegyszerűsítve a felhasználónak vázolnia kell;

- Mik a kiinduló adatok
- Milyen eredményt várunk az új rendszertől
- Az eredményhez milyen algoritmussal juthatunk el

Ez persze egy igen durva megközelítés. A rendszer feladata sokféle lehet;

- a bemenő adatok transzformálása más rendszerek felé
- az adatok tárolása, kimutatások készítése
- rendszeres adatszolgáltatás
- ezek kombinációja

A fentieken kívül minden olyan információt rögzíteni kell, ami befolyásolhatja a fejlesztés menetét, vagy a rendszer üzleti logikáját. Rögzíteni kell a külső tényezőket is, mint például;

- milyen jelenlegi rendszert használnak, ha van
- a jelenlegi rendszerből kell-e adatot átvenni (migráció)
- kik fogják használni a rendszert
- hányan fogják használni a rendszert
- milyen környezetben kell a rendszert megvalósítani
- mekkora adatmennyiséggel kell számolni

A felmérés jegyzőkönyve az egyik legfontosabb dokumentum.

A tervezést akkor érdemes megkezdeni, ha a felmérés teljeskörű. Ekkor már ismerjük az igényeket, és a kiindulási állapotot is. A tervezéskor a felmérés alapján meg kell tervezni, hogy az adatokat milyen strukturában érdemes nyilvántartani, milyen üzleti logikát kell kialakítani, illetve milyen felhasználói felületet kell készíteni. Ezek a legfőbb pontok, de persze az igényfelmérés határozza meg a részleteket.

A tervezéskor optimális esetben úgy érdemes eljárni, hogy maga a fejlesztés már csak a megvalósításra szorítkozzon, ne kelljen vissza-visszatérni a tervezéshez, vagy a felméréshez.

A rendszerterv készítésekor a felhasználó is tud segíteni, főleg a felhasználói felület tervezésekor, de ebben a fázisban neki már nem kell, hogy döntési jogköre legyen. A rendszerterv a fejlesztőknek készül, nem a felhasználóknak.

Ha elkészültünk a rendszer tervével, gyakorlatilag megkezdődhet a fejlesztés. Ezzel azonban nem áll meg a dokumentálás, hiszen még távolról sincs kész a rendszer.

A kivitelezés után következő dokumentum a tesztelési ütemterv. Ha elkészültünk egy modullal, vagy funkcióval, meg kell tervezni a funkció tesztelését. Ezt leginkább az a fejlesztő tudja elvégezni, aki készítette, hiszen ő tudja, mik lehetnek a buktatók. A tesztelés célja, hogy kipróbáljuk az összes lehetséges elágazást az üzleti logikában, vagyis a tervet is e-szerint kell megírni. Ez a terv leginkább a fejlesztői teszteléshez szükséges, de a feladat bonyolultságától függően a felhasználói teszthez is készülhet ilyen.

A végső átadáskor legkésőbb szükség van egy felhasználói dokumentációra. Ezt úgy érdemes megírni, hogy a felhasználó mindent megtaláljon benne, ha elakad. A dokumentum felépítése nem kötött, az egyetlen elvárás az, hogy a felhasználó könnyen átlássa. Többféle módszer létezik;

- Lehet a menü szerint haladni, tételesen felsorolva a funkciókat, az adatrögzítő felületeken a mezőket,
- vagy lehet a funkciók, használati sorrendek szerint haladni úgy, ahogy használni fogják.

Célszerű a funkciókat sok illusztrációval tarkítani, mert így a felhasználó könnyebben beazonosítja azt, hogy hol is tart.

Az elkészült fejlesztésről érdemes egy fejlesztői dokumentációt is készíteni függetlenül attól, hogy lesz-e folytatása a munkának. Ezt előre sosem tudhatjuk. A fejlesztői dokumentáció a karbantartást is nagyban segíti.

Ha a rendszer túl összetett, a karbantartáshoz szükséges lehet egy üzemeltetői dokumentáció is, ha más végzi az üzemeltetést, mint az „egyszerű” felhasználó.

Az eddigieken kívül még más dokumentumok is készíthetők igény szerint. Fontos még az is, hogy a kritikus, vagy jelentős pillanatokról jegyzőkönyvet, feljegyzést készítsünk, hogy később ezek ellenőrizhetők legyenek. Ilyen például a konzultációk jegyzőkönyve, vagy a módosítások, új igények kapcsán történt egyeztetések feljegyzése.

Utószó

Az itt leírtak nem általános aranyszabályok, ezek az én gondolataim. A fejlesztési projektekben ezeket a módszereket követem és hasznosnak találom.

A célom az, hogy segítséget nyújtsak másoknak. Ha valakinek sikerült utat mutatni, már elértem a célom.

Habár körültekintően próbáltam fogalmazni, előfordulhatnak hibák a gondolatmenetben. Ha a kedves Olvasó úgy gondolja, hogy tévedtem, szívesen fogadom a kritikát.

Sólyom-Nagy Péter

falco@vegacom.hu

2010. január