e-MedSolution – az alapok

Hadházi-Borsos Balázs, ISH Kft.

A publikáció bemutatja az e-MedSolution integrált kórházi rendszer alapvető felépítését és az olvasók megismerhetik a rendszer architektúráját. Ezek ismeretében a rendszer számos előnye is nyilvánvalóvá válik.

A cikk elsősorban azon felhasználóknak szól, akik már használták az e-MedSolution-t, vagy hallottak róla, és szeretnének vele kapcsolatban néhány technikai jellemzőt is megismerni.

Reading this article, you will find basic information about e-MedSolution, an integrated hospital informatics system. With the presentation of the basic architecture a few main advantages of this system will be presented, too. At the end you can find information related to dimensions of the system.

The article is primarily addressed for everyone who want to know a few technical details about e-MedSolution.

BEVEZETÉS

Az International System House Kft. az ezredforduló után célul tűzte ki – felismerve a web és az internet jelentőségét – egy olyan integrált kórházi rendszer (HIS) létrehozását, amely a magyar követelményeknek megfelel, ugyanakkor a korszerű web-es technológia előnyeit kihasználja. Az ISH Kft. ekkor már komoly eredményeket ért el az orvosi informatikában, több éves fejlesztői és üzemeltetői tapasztalattal rendelkezett a MedSolution termékkel kapcsolatban, amelyet számos hazai és néhány külföldi kórházban is teljes körűen használtak.

Mivel abban az időben a web-es technológiák még eléggé kiforratlanok voltak, csak az eddig felhalmozott mérnöki tudást lehetett újrahasznosítani, a rendszer minden egyes alkotóelemét egyesével le kellett programozni. Az elmúlt nyolc évben a rendszer folyamatosan bővült, számos új funkcióval, fejlesztéssel gyarapodott, újabb és újabb technikák lettek alkalmazva.

Az évek során az e-MedSolution-nel az ISH Kft. több hazai és külföldi bemutatón is megjelent. Ezek közül talán az egyik legrangosabb az évente Düsseldorfban megrendezésre kerülő MEDICA, amelyen az orvosi informatika területén ugyan a nyugat-európai, főleg a német piac dominál, de számos más terméket is látni lehet, ezért a világ minden részéről érkeznek érdeklődők. A felhasználói visszajelzéseket felhasználva, illetve a más termékekkel való összehasonlítás következtében a rendszer folyamatos átalakuláson megy át. A kezdeti próbálkozások óta a rendszer rengeteget fejlődött: ma már több hazai és külföldi kórházban az e-MedSolution jelenti az integrált kórházi rendszert.

Ebben a cikkben az Olvasó az e-MedSolution alapvető felépítésével ismerkedhet meg. Az alkalmazott technológia mellett – a teljesség igénye nélkül – a rendszer fontosabb komponenseit mutatjuk be, amit egy rövid összefoglaló követ az alkalmazott szoftver-fejlesztési módszerekről, valamint a rendszer méretéről.

A RENDSZERFEJLESZTÉS TECHNOLÓGIÁJA

Az ISH és az IBM közti partnerviszonynak köszönhetően nem csodálkozhatunk, ha az e-MedSolution fejlesztése IBM-termékekkel történik. Mivel a felhasználói verzió IBM WebSphere Application Server alatt fut, olyan fejlesztőeszközre volt szükség, amely ezzel nagyjából egyenértékű futási jellemzőkkel rendelkezik. Az IBM WebSphere Studio Application Developer fejlesztőeszköz mindazt tartalmazza, ami az e-MedSolution fejlesztéséhez szükséges: az Eclipse alapoknak köszönhetően a java, jsp, html stb. állományok kezelése, verziókövető-rendszerrel való együttműködése, a felhasználói futtató környezettel való egyenértékűsége mind pozitív tényező volt a megfelelő eszköz kiválasztásában.

Fontos tényező a fejlesztésben a verziókövető-rendszer, amely elősegíti a csoportmunkát, lehetővé téve a különböző fejlesztők munkáinak gyors és pontos integrálását egy közös verzióba, amelyből a felhasználói verziót el lehet készíteni, és amelyen integrációs teszteket lehet elvégezni. Mindezen tulajdonságai mellett a Rational ClearCase segítségével évekre visszamenőleg meg lehet mondani, hogy a rendszer alkotóelemét képező bármely forrásállomány mikor került a rendszerbe, pontosan mikor és ki módosította. A ClearCase-ben minden egyes forrásállománynak külön verziója van, némelyiknek a sok módosítás következtében a verzió-fája elég kusza, a kezdő fejlesztőket minden bizonnyal megrémiszti, viszont hatalmas előny, hogy a fejlesztők úgy dolgozhatnak a különböző fejlesztéseken, hogy közben az integrálói verzió – és ezáltal a kiadásra szánt végfelhasználói verzió – érintetlen marad.

A rendszer tervezésénél kiemelten fontos szempont volt, hogy az ISH aktuális ügyfelei alacsony bevezetési/átállási költségek mellett és rugalmasan, több lépcsős technikával tudják majd az e-MedSolution-t bevezetni. Ennek megfelelően a Progress 4GL-ben íródott MedSolution és a web-es e-MedSolution képes ugyanazon az adatbázison egymás mellett futni úgy, hogy az egyik rendszer által létrehozott/módosított adatokat a másik rendszer maradéktalanul kezelni tudja. Ez bevezetésnél, az egyik rendszerről a másikra való átállásnál hatalmas előny, mivel egyrészt az átállás időben jobban ütemezhető, a felhasználók oktatása rugalmasabban tervezhető, nincs szükség az adatok migrációjára, áttöltésé-

re, konvertálására, és végül nincs szükség új adatbázis-kezelő rendszerre sem.

A RENDSZER FELÉPÍTÉSE

Az e-MedSolution egy J2EE alkalmazás, a klasszikus jsp-servlet technológiát használja. Ez leegyszerűsítve azt jelenti, hogy a kliens oldalon html állományok jelennek meg egy internetes böngészőprogramban, az itt lévő link-ek, ikonok és gombok egy-egy kérést generálnak az applikációs szerver felé, amely a kéréseket a megfelelő servlet-eknek (java objektumoknak) delegálja. A servlet feldolgozza a kérést, lefuttatja a neki megfelelő üzleti logikát, és a választ a jsp-nek továbbítja, amely előállítja a html oldalt, amit visszaküld a felhasználó böngészőjének.

Ennek az architektúrának az a sajátossága, hogy vékony-klienses alkalmazást tesz lehetővé, azaz a felhasználói terminálon gyakorlatilag nem fut semmilyen rendszerkomponens, mindössze egy böngészőprogram. Tény, hogy az újabb böngészőprogramok csak újabb operációs-rendszereken futnak, amelyeknek viszont a hardver-igényével is számolni kell. Mindemellett az e-MedSolution kliens-oldali hardver-igénye még mindig jóval kisebb, mint a szintén népszerű Windows-os vastag-klienses HIS rendszereké, ráadásul azokkal szemben megvan az az előnye is, hogy új fejlesztéseket, hibajavításokat összehasonlíthatatlanul könnyebb telepíteni, rendszerkarbantartást végezni viszonylag egyszerű, mivel az üzemeltetőknek egyetlen gépen – a szerveren – lévő szoftvert kell csak módosítani, és nem kell aktualizálni a kórház összes számítógépét.

Mivel a jsp-servlet architektúra kérések kiszolgálására képes, nincs az e-MedSolution-ban egy állandóan futó háttérfolyamat. Az applikációs szerver, az IBM WebSphere fut a háttérben a kórház szerverén, és folyamatosan várja a kliensektől érkező kéréseket. A rendelkezésre-állása és a megbízhatósága nagyságrendekkel meghaladja egy egyszerű PC hasonló jellemzőit.

A rendszer tervezésénél és fejlesztésénél arra törekedtünk, hogy a megfelelő rendszerkomponensek létrehozásával az egyes funkciók fejlesztésekor már ne kelljen a jsp-servlet architektúrával foglalkozni, elegendő kizárólag az üzleti logika implementálására koncentrálni. Ezt természetes módon segíti elő a java programozási nyelvben az objektum-orientált megközelítési mód: az egyes objektumok újrafelhasználhatóak, bizonyos elemeik az újrafelhasználás következtében módosíthatóak, így megváltozott működést eredményeznek – mindezt úgy, hogy az eredeti objektum működése, ezáltal az eredeti funkció nem sérül. Az objektumoknak ezt a szerveződését nevezzük osztályhierarchiának.

Többrétegű technológiai megoldás

Az e-MedSolution rendszer több rétegből áll. Az adatbázis-kezelést a DAO-réteg valósítja meg (Database Access Object). Ezek az objektumok definiálják, hogy az adatbázisból hogyan kell kiolvasni az egyes funkciókhoz tartozó adatokat. Az adatbázisban lévő azon adatok java oldalon való tárolására, amelyek tranzakcióban vesznek részt, a DataBaseObject objektum (DBO) leszármazottait használjuk. Ezek az objektumok többé-kevésbé az adatbázisban lévő táblák egy-egy rekordjának felelnek meg. Az analógiát az indokolja, hogy módosításnál tranzakcióra és az adatok megfelelő zárolására van szükség az adatbázis integritásának megőrzése érdekében, így az adatbázis-kezelő rendszereknél általánosan használt zárolási mód van leképezve, amelyben a legkisebb zárolható egység egy adattábla egy rekordja. Értelemszerűen a DAO és DBO objektumok újrafelhasználása meglehetősen magas: például a rendszerben bárhol jelenik meg egy páciens-adat, azt ugyanaz a DBO_Pbasic ábrázolja, és ugyanaz a DAO_Pbasic olvassa ki adatbázisból, majd szükség esetén írja oda vissza (Pbasic = Patient Basic Data, azaz páciens alapadatok).

Az üzleti logika minden esetben egy Command-ban van implementálva. A Command és segédobjektumai (funkciószintű megjelenítés vezérlő, ellenőrző-objektumok stb.) felelősek a specifikációban rögzítettek teljesítéséért. Mivel a Command tulajdonképpen egy funkciónak felel meg, könnyen belátható, hogy a Command-nak a rendszerben történő újrafelhasználása meglehetősen alacsony, hiszen a funkciók általában nem ismétlődnek. Természetesen itt is van kivétel, mivel a rendszer sok hasonló funkciót tartalmaz, így például az összes teljesítőhely munkalista üzleti logikája külön Command-ban van, amelyek egy általános munkalista-objektum alá csoportosulnak.

Annak érdekében, hogy az üzleti logikából minél többet lehessen újrafelhasználni, ezen belül is szinteket különböztetünk meg. Az adatok felhasználói felületen való megjelenítésére egy speciális objektumhierarchiát hoztunk létre – itt éppen úgy objektumként van ábrázolva egy egyszerű címke is, mint a legbonyolultabb lista, fa-szerkezet vagy akár az egész űrlap. Ennek a technikának köszönhetően, amennyiben a felhasználói visszajelzés alapján a képernyő egy elemét módosítani kell, lehetőségünk van arra, hogy egy, az új igényeket megvalósító felület-objektumot hozzunk létre, és ezzel lecseréljük a kifogásolt komponenst. Ez azt eredményezi, hogy a rendszer meglévő komponense nem módosul, ami azért fontos, mert az adott komponens egyéb felhasználási területét a bevezetett módosítás semmilyen formában sem érinti, azaz más funkciók nem sérülnek, így egyrészt nagyobb biztonsággal lehet az új fejlesztést integrálni, másrészt kevesebb tesztelés szükséges.

Végül a felhasználóhoz legközelebb lévő réteg tartalmazza a dizájnt, amelyet a html-hez kapcsolódó css-ek (Cascading Style Sheets) és képek alkotnak.

A rendszer sajátossága, hogy tartalmaz egy saját fejlesztésű képernyőtervezőt annak érdekében, hogy a felhasználói felületet és a hozzá tartozó üzleti logika egy részét ne kelljen kézzel megírni. A képernyőtervező a rendszer többi funkciójához hasonlóan a fent felsorolt objektumokból épül fel, ez is egy funkció a rendszerben, mint például egy betegfelvétel. A képernyőtervezőben egy interaktív felületen lehet összeállítani a képernyőt, meghatározva az egyes képernyőelemek attribútumait, majd ebből a tervező

jsp és java forrásállományokat generál, amelyek a rendszerben módosítás nélkül felhasználhatóak. Természetesen amennyiben az üzleti logika bonyolultabb, lehetőség van a megfelelő osztályhierarchia használatával a generált kód bizonyos részeinek felüldefiniálására.

Objektum kezelés

A fent felsorolt rétegek mellett a rendszer általános rendszerobjektumokat is tartalmaz, például a servlet-eket a kérések kiszolgálásra, tranzakció-kezelő objektumot, amely felelős az adatbázis integritásának megőrzéséért, felhasználóhoz kapcsolódó, úgynevezett session-adatokat tartalmazó objektumokat stb.

A rendszerben nagy fontosságot tulajdonítunk a cache-eknek – számos egy családban lévő objektumhoz egy cache tartozik. Ennek a megközelítésnek az a hatalmas előnye, hogy az objektumot (például kódtábla) nem kell közvetlenül adatbázisból felolvasni minden egyes felhasználáskor, elegendő a cache-től elkérni. Az, hogy az objektum éppen a memóriában van, vagy tényleg adatbázisból kell felolvasni, a cache feladata, az üzleti logika (Command) szempontjából csak az a fontos, hogy az adott objektumot megkapja. Ezzel a technológiával a rendszer ugyan több memóriát foglal, viszont gyorsabban működik, mert a relatív lassú adatbázismműveletek helyett az adatot közvetlenül a számítógép memóriájából elérhetjük. Itt fontos megjegyezni, hogy ezt az alkalmazott szerver-technológia teszi lehetővé: a programok nem a kliens gépeken futnak, hanem a szerveren, így a több felhasználó által futtatott különböző funkciókhoz tartozó közös adatok adatbázisból való felolvasása egyszeri processzor- és adatbázismővelet-költségként jelentkezik. Ugyanakkor, mivel a cache az összes felhasználó számára közös, az összes adat a szerver számítógépen maximum egy példányban van tárolva (és nem minden kliens gépen), így a rendszer össz-memóriaigénye nem túl nagy (néhány száz megabájt), ráadásul a szerverek eleve olyan memória-kapacitással rendelkeznek, hogy az e-MedSolution cache-ek nem terhelik le a számítógépet.

Szintén az alkalmazott technológiának köszönhetően az e-MedSolution hatékonyan kihasználja az úgynevezett Connection Pooling-ot, azaz az adatbázis-kapcsolatok központi menedzselését és újrafelhasználását – alkalmazkodva az eddigi fogalmakhoz, az adatbázis-kapcsolatok központi cache-elését. A felhasználói interakciónak megfelelő üzleti logika kikeresi az adatbázisból az adatokat, és megjeleníti azt a felhasználói felületen. A Connection Pool a rendszer indításakor létrehoz néhány adatbázis-kapcsolatot, ezeket eltárolja a memóriában, majd amikor egy üzleti logikának adatbázis-műveletre van szüksége, akkor a kapcsolatot odaadja, majd a művelet végén visszakapja. Mivel a rendszerben lévő adatbázis-műveletek relatív gyorsak, egy néhány tizedmásodperc alatt befejeződnek, ezért az adatbázis-kapcsolat hamar felszabadul, és más üzleti logika rendelkezésére áll. Ennek köszönhetően relatív kisszámú adatbázis-kapcsolat elegendő a rendszer működéséhez, mivel amennyiben adott pillanatban nincs szabad adatbázis-kapcsolat, az üzleti logika gyakorlatilag észrevehetetlenül kis ideig várakozik, majd adatbázis-kapcsolathoz jut, és folytatja működését. A Pool-ban lévő adatbázis-kapcsolatok számának paraméterezésével elérhető, hogy mindig legyen egy néhány szabad adatbázis-kapcsolat, azaz egyetlen üzleti logika se várakozzon. Ez a technológia azt eredményezi, hogy a rendszerbe bejelentkezett minden felhasználónak nem kell saját egyedi adatbázis-kapcsolatot fenntartani, jóval kevesebb is elegendő, tehát a kórháznak kevesebb adatbázis licencet kell vásárolnia.

Mindezek mellett a rendszer alapját képezik az állapotgépek, azaz azok az objektumok, amely nyilvántartják, hogy a felhasználó egy adott munkafolyamatban, egy adott funkció megvalósításában pontosan hol tart. Ezek az objektumok vezérlik a rendszer működését, például csak akkor engedélyezik az adatbázisba való rögzítést, ha azok ellenőrzése és jóváhagyása már megtörtént stb.

RENDSZERTERVEZÉS

Egy felhasználói igény megvalósítása a következő jól körülhatárolható lépésekből áll:

- Szoftverspecifikáció, melynek során az ismert felhasználói igények rögzítésre kerülnek.
- Szoftvertervezés és implementáció, melynek során az igényeket teljesítő, és a rendszerbe illeszkedő szoftverkomponens megvalósításra kerül.
- Szoftvervalidálás, amely biztosítja, hogy az ügyfél azt kapja, amit szeretne.
- Szoftverevolúció, azaz az átadást követően az újabb felhasználói igények szerint a szoftver aktualizálása.

Az egyszerűbb funkciók esetében ezek a lépések jól elhatárolhatóak. Az ügyfél jelez egy igényt, amelyet az e-MedSolution tanácsadók egyeztetnek és specifikálnak. Ezután a rendszer- és szoftvertervezés következik, amikor a rendszer egészére rálátással rendelkező személyek (rendszertervező-fejlesztő) definiálják azokat a technikai követelményeket és meghatározzák azokat a kapcsolódási pontokat, amelyek ismeretében az implementáció és fejlesztői teszt elvégezhető. Ezt követi egy integrálási folyamat, amikor az adott fejlesztői környezetből egy közös verzióba, azaz átadható állapotba kerül a modul. Az integrációs teszt a modul saját hibáin túl a többi rendszerkomponenssel való kapcsolódási hibákat is felderíti, így azok javítására még a termék átadása előtt lehetőség van. Ezután következik a kész szoftver átadása. Ezek a lépések a klasszikus szoftverfejlesztési modell, a vízesésmodell alkotóelemei.

A bonyolultabb igények teljesítése érdekében egyéb szoftverfejlesztési modellt is alkalmazunk, például az evolúciós fejlesztést, amikor a definiált igények alapján létrejön egy prototípus, amelyet az ügyfél megtekint, pontosítja az igényeit, erre módosul a szoftver, egészen addig, ameddig az ügyfél elégedett nem lesz. Ez a modell abban az esetben alkalmazandó, amikor az igényelt funkció annyira összetett, hogy a felhasználó nem tudja részletesen és pontosan elmondani a követelményeket.

AZ e-MedSolution SZÁMOKBAN

Az ISH az évek során számos ügyféllel került kapcsolatba, melynek eredményeképpen az e-MedSolution rengeteg, felhasználói igényre készített fejlesztést tartalmaz. Mivel ezek a felhasználói igények folyamatosan változnak, ráadásul a rendszer a szintén állandóan változó jogszabályi követelményeknek is mindig meg kell, hogy feleljen, az e-MedSolution a fejlesztés nyolc éve alatt 2000-nél (azaz kettőezernél) is több verzió-módosítást élt meg. Az e-MedSolution több mint 1100 képernyőt tartalmaz, amelyeket összesen 1600-at meghaladó funkció jelenít meg. A kb. 10 200 java forrásállomány (egy forrásállomány egy java objektumnak felel meg) összesen több mint 1 000 000 (azaz egymillió) programsorból áll.

Annak ellenére, hogy az e-MedSolution-t számos kórházban teljes körűen használják, a rendszer fejlesztése korántsem fejeződött be. Az ISH folyamatosan regisztrálja az újabb és újabb fejlesztési igényeket, amelyeket teljesítve – reményeink szerint – a felhasználók az e-MedSolution-t mind nagyobb és nagyobb megelégedéssel fogják használni.

A SZERZŐ BEMUTATÁSA

Hadházi-Borsos Balázs
1994–1998: egyetemi képzés – „Babeş – Bolyai” Tudományegyetem (Kolozsvár, Románia), Matematika és Informatika Kar, Informatika szak
1998–1999: Master of Science továbbképzés mesterséges intelligencia szakon – „Babeş – Bolyai” Tudományegyetem (Kolozsvár, Románia), Matematika és Informatika Kar, valamint „Johannes Kepler” Tudományegyetem (Linz, Ausztria)
2008-tól: PhD képzés – Debreceni Egyetem (Debrecen, Magyarország), Informatikai Kar
1999-től foglalkozik orvosi informatikával. Az e-MedSolution fejlesztésében a kezdetektől aktívan részt vett, jelenleg vezető szoftverfejlesztőként, rendszerszervezőként és tanácsadóként segíti az e-MedSolution fejlesztését, üzemeltetését és új kórházakban történő bevezetését. Célkitűzései között szerepel a szoftver-ergonóma szempontjait figyelembe véve az e-MedSolution kezelhetőségének javítása.

Folytatás a 25. oldalról

A szkizofrénia kezelésében sajátos nehézséget jelent, hogy a betegek nehezen fogadják el a betegség tényét, a hosszantartó terápia szükségességét. A mindennapi klinikai gyakorlatban kiemelt jelentősége van tehát a páciens együttműködését javító terápiás módszereknek, a bizalmon alapuló orvos-beteg kapcsolatnak és kommunikációnak. **Dr. Xavier Amador,** a Teachers College Columbia University (USA) klinikai pszichológia professzora előadásában ismertette az általa kidolgozott ún. LEAP (Listen-Empathize-Agree-Partnership) módszert. Bemutatta, hogyan alakíthatunk ki bizalmi légkört akkor is, ha betegünk vagy családtagunk meg van győződve arról, hogy nincs szüksége orvosi segítségre. Amador professzor módszerét a Nem vagyok beteg, nincs szükségem segítségre! című könyvében adta közre, amelynek magyarországi bemutatójára a 10. Szkizofrénia Akadémián került sor. „Rájöttem, hogy azok a tudományos eredmények, amelyeket én már jól ismertem, sok olyan emberhez még nem jutottak el, akiknek pedig nagy szükségük volna erre a tudásanyagra. Ezért írtam ezt a könyvet.” – írta könyve előszavában a szerző, aki részben pszichológusként, részben családtagként megtapasztalta a szkizofrénia kezelésének nehézségeit. A könyvbemutatón kifejtette: a módosult agyműködés következtében a szkizofrénia gyakran, az esetek nagyjából felében jár együtt a betegségbelátás teljes vagy részleges hiányával – anozognóziával -, ami értelemszerűen azt eredményezi, hogy a páciens szükségtelennek ítéli a kezelést. Kérdésünkre válaszolva dr. Amador elmondta, hogy feltehetően az anozognózia egyéb kórképekben, például bipoláris depresszióban, alkoholizmusban és kényszerbetegségben is előfordul, e tekintetben jelenleg aktív kutatások zajlanak világszerte. A LEAP módszer a szkizofrénián kívül számtalan más probléma megoldásában is hatásosnak bizonyult. Dr. Amador könyve első kiadásának megjelenése óta több ezer embert – családtagokat és egészségügyi szakembereket – tanított meg a módszerre, hogy miként győzzék meg a mentális betegségben szenvedőket a kezelés elfogadásáról. Dr. Amadorról és a LEAP szemináriumokról a www.XavierAmador.com weboldalon található további információ.

Munkatársunktól