- Koplányi Krisztián -

Szoftverek felépítésének átgondolása az alapoktól

Dátum: 2013. 05. 19.

1. FELHASZNÁLÁSI FELTÉTELEK

2. ELŐSZÓ

2.1. Bevezető rizsa

A számítástechnika fejlődésével a szoftverek felépítése, tudása, működése, felhasználási területe folyamatosan változik. A számítástechnika „(h)őskorában” lerakott alapok a **Z-generációs** felhasználók (digitális bennszülöttek) szemében „régészeti leletnek” tűnhetnek.

A rövid és egyszerű programkód, kis fájlméret – sajnos – manapság már nem szempont. Olyannyira, hogy a több évtizedes múltra visszatekintő programok (pl. CAX felhasználási területen) ősrégi kódokat görgetnek magukkal. Ennek hozománya, hogy e szoftverek nagy része régi architektúrákra épül, így maguk a szoftverfejlesztők sem feltétlen merik kigyomlálni és kiváltani az elavult részeket.

A szoftverek és hardverek útja régóta kettévált, és folyamatosan távolodnak egymástól. Utoljára talán a C64 -es időkben (demoscene nemzedék) lehetett optimális a helyzet a helytakarékosság, valamint az erőforrás kihasználtság szempontjából. A mai programok a mai hardvereken nem feltétlen futnak jobban, gyorsabban mint a 30 éves programok az akkor aktuális gépeken.

Az okos-telefonok, táblagépek elterjedésével a személyi számítógép fogalma is átértékelődött, emellett a „felhő” alapú szolgáltatásokkal az adattárolási megoldások szemlélete is megváltozott.

Külön említést érdemel a **felhasználói élmény** fogalma. Már nem elégszünk meg azzal, hogy csupán funkcionálisan működik valami; az is fontos, hogy az adott eszközt külön élmény legyen használni (eye-candy effektus).

2.2. A lényeg rövidebben

Általános célú feladatokra (szövegszerkesztés, képnézegetés/szerkesztés, stb.) számtalan program íródott, némelyik célszoftverre viszont alig van alternatíva.

Még mindig vannak olyan feladatok, amelyek megoldására érdemes új szoftver fejlesztésébe fogni. Ezen felül az egyes régi alkalmazásainkat érdemes lehet totálisan újraírni a mai kor követelményeinek megfelelően.

E jegyzetben arra kívánok rámutatni, hogy a program fejlesztőinek (a leendő felhasználókkal együtt) milyen szempontokon kell átrágniuk magukat, hogy a végeredmény egy jól használható munkaeszköz legyen.

TARTALOMJEGYZÉK

3. ALAPELVEK

3.1. Multi-platform

A jelenleg aktuális operációs rendszerek figyelembe vételével keresztplatformos programokat célszerű alkotni. Ehhez megfelelő programozási nyelvet, fejlesztői környezetet, widget eszköztárat, verziókövető- és hibabejelentő rendszert javasolt választani már a projekt legelején.

Támogatandó platformok

- BSD (Free-, Net-, stb.);
- Linux alapú:
  - tar.gz (Slackware, Frugalware, stb.);
  - deb (Debian, Ubuntu, stb.);
  - rpm (Red Hat/Fedora, OpenSuSe, stb.);
- Android;
- OS X / iOS;
- Windows.

Lehetséges...

programozási nyelvek:

- C++, Fortran, Free Pascal, Java, Perl, Python, Ruby;

IDE -k (integrált fejlesztői környezetek):

- KDevelop (C/C++, Qt), Eclipse (Java), Geany (a legtöbb programnyelvet támogatja), Lazarus (Free Pascal, Delphi), Netbeans (Java), OpenJDK (Java), Python IDE (Python, Ruby), Pida (Python);

widget eszköztár (grafikus megjelenéshez):

- Qt, Glade, Lazarus, SPE (Python), wxFormBuilder (C++, Python);

verziókövető rendszerek:

- Bazaar, Cervisia, KDESvn, git.

3.2. Nyílt forráskódú szemlélet

„KISS” elv (tartsd bután, egyszerűen!)

- Jól strukturált, moduláris programkód, mely tartalmaz kommenteket (más szakember számára is egyértelmű legyen);
- Többnyelvűség támogatása:
  - karakterkódolás (pl.: UTF-8);
  - nyelvenként külön nyelvi fájlok (pl.: mint a po fájlok);
- Testre szabható GUI (grafikus felhasználói felület):
  - skinnelhető, különböző előtét programok;
  - szemkímélő színösszeállítás (a téma színvilága illeszkedjen a rendszer alapértelmezett beállításaihoz, egyéni színösszeállítás készítésének lehetősége);
  - átlátható, logikus menürendszer;
  - makrózható parancsok;
- Fájltípusok, fájlkonverziók, mentés:
  - fájltípusok definiálása (The Document Foundation -el való együttműködés);
  - szabványos fájltípusok és fájlkonverziók támogatása (pl.: odf, pdf);
  - mentés, szinkronizálás különböző „felhős” szolgáltatásokkal (pl.: iCloud, Dropbox, Ubuntu One, LogMeIn, SkyDrive);
- Más nyílt projektekre való épülés (pl: MySQL, MariaDB).

3.3. Végfelhasználói licenc választása

Szabadon használható programok esetén: GNU/GPL, LGPL, stb. licenc választása.
Fizetős felhasználás esetén: egyéni licenc, valamint:

3.3.1. Tanulói verzió (≠30 napos tesztverzió)

- Időkorlát nélküli, de üzleti felhasználásra nem használható.
- Ennek tényét csak szolidan közölje (adott dokumentumban csak az első oldal fejlécében/láblécében legyen feltüntetve).
- Rendelkezzen az üzleti verzió funkcionalitásával, valamint nyissa meg és kezelje azok fájljait.

Felvetés

Ha a program erősen épít különböző adattárakra (amelyeknél a friss verzió versenyelőnyt jelent), akkor a tanulói változatba ezen adatbázisok előző évi változata lenne alapértelmezetten beépítve. Az üzleti verzióban „csak” az adatbázisok frissességét kellene megfizetni, valamint a programhoz nyújtott technikai támogatást.

3.3.2. Üzleti verzió

- Legyen kompatibilis a tanulói változat fájljaival.
- Moduláris felépítés (az egyes szakmodulok külön-külön megvásárolhatóak).
- Egyéni és/vagy hálózati licenc biztosítása, valamint a licenc megadásával a tanulói változatból automatikusan üzleti módba váltson a program (így elég operációs rendszerenként 1-1 db telepítő fájl).
- Technikai támogatás lehetőségének biztosítása.
- Betanítási lehetőség.

3.3.3. A program elérhetősége

- A projekt honlapjáról legyen elérhető a program telepítője (operációs rendszerenként külön-külön); valamint a súgó, segédletek, példák (offline pdf, hivatalos/közösségi youtube videók).
- A különböző linux disztribútorokkal kapcsolattartás, tárolók (repository) biztosítása.

3.4. Az alkalmazás logikai felépítése

Ha felmerül igény egy új program elkészítésére (amely egy piaci rést töltene be), akkor egyértelműen és világosan tisztázni kell az alkalmazás funkcióit és működését, amelyeknek valós piaci igényeken kell alapulniuk.

3.4.1. Funkciók és működés

Ha egy meglévő szoftvert szeretnénk teljesen az alapoktól újraírni, akkor fontos összegezni:

- a meglévő alkalmazásunk előnyös és hátrányos tulajdonságait;
- a következtetések tükrében tisztázni az elvárásokat az új programmal szemben.

A régi DOS alapú alkalmazások funkciójuk tekintetében betöltik ugyan a feladatukat, de a „felhasználói élmény” szempontjából nem felelnek a mai kor felhasználóinak. A keresztplatformos szemlélet, modularitás, bővíthetőség, konfigurálhatóság miatt érdemes átgondolni, milyen új értékeket adjunk hozzá a programhoz.

3.4.2. Adatbázisok

Amennyiben a program különböző adattárakkal, elemtárakkal, táblázatos jellegű adatokkal operál, nyilván szükség lesz adatbázis-kezelő programra.

Az adatbázis kezelés során fontos (adatok indexelése, keresése, lekérdezése szempontjából), hogy megfelelő adatstruktúrát definiáljunk (pl. : mik szerepeljenek a táblázat mezőiben, az egyes rekordok hogyan legyenek azonosítva, kulcsok megadása).

A felhasználóknak legyen lehetőségük saját adattárak létrehozására. Új adatbázisok létrehozásához, illetve a meglévőek szerkesztéséhez/bővítéséhez szükség van grafikus előtétprogramra (front-end).

3.4.3. GUI (grafikus felhasználói felület)

Fontos tisztázni, hogy az alkalmazás menürendszere hogyan épüljön fel (fájl, szerkesztés, nézet, beszúrás, eszközök, ablak, súgó, stb.). Az egyes funkciókat ésszerűen kell csoportosítani.

A gyakran használt parancsoknak külön indítóikonok készítése (amik lehetőleg vektorgrafikus piktogramokból álljanak, pl.: svg). Az ikonokra egérrel rámutatva jelenjen meg buborékban annak szöveges magyarázata (pl.: fekete alapon fehér betűvel).

Sablonok használatának támogatása. Definiálásukhoz űrlap készítése.

A szoftvernek külsőre jó benyomást kell keltenie, amellyel „öröm dolgozni”.

3.4.4. A program lelke

Ez az előző két rész között helyezkedik el. Különös ismérve, hogy normál földi felhasználó ebbe a részbe nem kotnyeleskedik bele, csak elvárja, hogy jól működjön. Ez a programozó játszótere („easter eggs” elbújtatása).

Megjegyzés

Ahhoz, hogy a szoftver egyes részei „harmóniában” legyenek egymással,

- az adatbázis gurunak;
- a programozó zseninek;
- a grafikus megjelenést készítő „művész léleknek”;
- a tesztelést végző felhasználóknak

megfelelően össze kell dolgozniuk egymással.

---

4. ZÁRSZÓ

Jogosan vetődhet fel az olvasóban a kérdés: a lökött, (programozói szempontból) dilettáns szerző vajon milyen programok írására/újraírására buzdíthatja a programozókat?

A meglévő DOS -os rettenetek leváltását sürgetném első sorban, amelyek funkcionálisan működnek ugyan, de megakadályozzák/hátráltatják a régi hardver- és szoftverkörnyezetről való migrálást.

A különböző célszoftverekben általában közös, hogy nem támogatják a fentről lefelé építkezés (top-down design) gondolkodásmódját, ezáltal a felhasználók tevékenysége nem a kreatív problémamegoldásra irányul, hanem a rabszolgamunkára korlátozódik. Célszerű lenne a kreatív munkavégzés elősegítése/támogatása; amelynek elengedhetetlen kelléke a parametrikus, asszociatív, integrált szoftverek – mint munkaeszközök – biztosítása.

Néhány lehetőség

- költségvető;
- számlázó;
- raktárkészlet kezelő;
- könyvelő;
- vállalatirányítási rendszer;
- egészségügyi nyilvántartó rendszer (házi orvosok, kórházak számára);
- különböző hivatalos szervek nyilvántartási rendszerei (föld-, „Polgár Jenő” hivatal, rendőrség).

Elképzelhető olyan megoldás is, hogy a fent felsorolt első 4 alkalmazás egy vállalatirányítási rendszer 1-1 külön almodulja legyen. Ebben az esetben bizonyos paramétereket érdemes „megjelölni”, hogy azok más modulokban asszociatív módon felhasználhatóak legyenek:

példa:
Egy költségvetésben pl. szerepel a projekt megnevezése, munkaszáma, megrendelő neve, ajánlati összeg. A számlázó program ezeket az adatokat automatikusan átszipkázhatná a költségvetésből.

Megjegyzés

Néhány év múlva elképzelhető, hogy ez a dokumentum is az „internetrégészek” megmosolyogtató leletévé fog válni. Bárcsak már ott tartanánk!
