



## 2.7 FEJLESZTÉS ÉS MŰKÖDTETÉS

Ebben a témakörben tárgyaljuk az informatikai rendszerek létrehozásához (tervezés, implementálás), valamint a teljes életciklusuk követéséhez használt módszereket és eszközöket, beleértve a munkafolyamatok és az innovációs folyamatok technológiai és szervezési támogatását.

A rendszerek – elsősorban a szoftverkomponenseik miatti – növekvő bonyolultságának kezelése a számítástechnikai rendszerek létrehozásának kezdetétől fogva (ld. a 60-as évek közepén jelentkező „szoftverkrízis” jelenséget) alapvető problémát jelentett. A rendszerfejlesztés technológiai és munkaszervezési eszközeinek területén elért jelentős eredmények ellenére az egyre összetettebb és kritikus feladatokat ellátó rendszerek határidőre való elkészülése és nagy megbízhatósággal való működtetése továbbra sem felel meg a felhasználók – egyre növekvő – kívánalmainak. A bonyolult szoftvereket (pl. operációs rendszereket) a kidolgozójuk termékként forgalmazza, azok működéséért ill. továbbfejlesztéséért a felelősséget elvileg vállalja, ami azonban a gyakorlatban csak nehezen érvényesíthető. A bonyolultság növekedésével ez egyre inkább tarthatatlanná válik és (többek között) ennek következtében jelennek meg a „nyílt forráskód” elvén alapuló megoldások, amikor a termékek karbantartását és követését a felhasználók széles közössége kívánja elvégezni.

A szoftvertermékek felhasználásával kapcsolatos követelmények gyors és dinamikus változása párosulva a szoftver kézenfekvő és könnyen kivitelezhető módosításának igényével olyan új fejlesztési módszerek és üzleti modellek kialakulásához vezet, amelyekben a termékek előállításával szemben a szolgáltatás jellegű feladatok kerülnek előtérbe. Ez egyrészt abban a paradigma váltásban jelenik meg, hogy a hagyományos „programozás” helyébe a rendszerek meglévő – sok esetben WEB-SZOLGÁLTATÁSOK formájában az Interneten található – elemekből való összeépítése lép. Másrészt a különböző alkalmazás- és infrastruktúra-üzemeltető cégek megjelenése, valamint a szoftver- és hardverelemek egységes informatikai szolgáltatásokba történő beintegrálása mögött is ez áll.

### Részterületek fejlődése



### 2.7.1 Tervezési-elemzési nyelvek és támogató eszközeik

Fokozódik a konvergencia az objektum-orientált elemzési és tervezési nyelvek, (melyek tényleges szabványa ma az UNIFIED MODELING LANGUAGE – UML), az ún. ÜZLETIFOLYAMAT-ELEMZŐ (BUSINESS PROCESS ANALYSIS – BPA) eszközök, valamint az adatbázistervezést támogató eszközök között. Ez egyre inkább lehetővé teszi a felhasználók számára, hogy ezeket az egymáshoz kapcsolódó és egymást kiegészítő kompetenciákat minél jobban kihasználják.

A legjobb BPA, UML és adatbázis-tervező eszközök integrációja már jelenleg is folyamatban van. Az UML 2. verziója fogalmak szélesebb körét támogatja nagyobb következetességgel, és szűkíti a rést az üzleti és a műszaki követelmények között, azonban az új elemek szoftver támogatásának bizonytalanságai még késleltetik a terjedését.

Annak az igénynek a kielégítésére, hogy az alkalmazásfejlesztés teljes életciklusában a minőség és integráltság érdekében az üzleti és technológiai modellek 2008-ra átláthatóvá és átjárhatóvá váljanak, várhatóan az ÜZLETIFOLYAMAT-ELEMZŐ (BPA), az UML alapú modellező és az adatbázis tervező eszközök – a versenyben megmaradó gyártóknál – egyetlen integrált eszközkészletté alakulnak át.

### 2.7.2 Nyílt forráskódú szoftverek (Open Source Software)

A NYÍLT FORRÁSKÓDÚ (OPEN SOURCE SOFTWARE – OSS), licencköteles termékek piaci részesedése növekszik, és ez több szoftverpiacon is nyomást gyakorol a hagyományos megoldásokra. Emellett a nyílt forráskód fejlesztésének bevált gyakorlata alapvető változásokat kényszerít ki az informatikai ipar szoftverfejlesztéshez, -elosztáshoz és -támogatáshoz való hozzáállásában. Ezek a hatások globális méretekben játszódnak le, és hamarosan megváltoztathatják a piaci erőviszonyokat.

Az OSS-MODELL hatással van a szoftvert kifejlesztésének módjára – a hangsúlyt a felülről jövő ellenőrzéssel szemben az együttes (collaborative) erőfeszítésekre helyezve. Ez befolyásolja a szoftverek használati módját, előtérbe helyezve a szolgáltatás-alapú kereskedelmi modelleket a szoftver licenccijakkal szemben (azaz a felhasználó nem magáért a szoftver termékért fizet, hanem az annak felhasználásához szükséges oktatási, testreszabási, karbantartási, üzemeltetési stb. szolgáltatásokért). Végül ez a modell a szoftver támogatására is hatással van, mert a hangsúlyt a minőség biztosítása és evolúciója terén közvetlenül a felhasználói közösségre helyezi.



### 2.7.3 Modellvezérelt és szolgáltatásorientált alkalmazásfejlesztés

A modellvezérelt alkalmazásfejlesztés a fejlesztés tárgyát képező terület további gépi feldolgozást lehetővé tevő, formális leírásából, azaz modelljéből indul ki (MODEL-DRIVEN ARCHITECTURE – MDA)

Manapság az ún. „szolgáltatások” válnak a szoftverek komponensekre való felosztásának elsődleges egységeivé. Az – egymáshoz nyílt és szabványos felületen kapcsolódó – szolgáltatásokból a korábinál jóval rugalmasabb architektúrával rendelkező szoftvereket lehet létrehozni. (SERVICE ORIENTED ARCHITECTURE – SOA)

Az architektúratervezésre, GYORSFEJLESZTÉSRE és modellezésre alapuló megközelítéseknek a szolgáltatásorientált alkalmazásfejlesztés során való együttes alkalmazásával jelentős termelékenységgjavulás érhető el. Ennek eredményei várhatóan nyílt és széles körben használt keretrendszerek, ill. platformok lesznek, amelyeket mind az eszkögyártók, mind a nyílt forráskódot támogatók, mind pedig a végfelhasználók közösségei elfogadnak.

A Gartner előrejelzései szerint 2008 körül azok az IT-szervezetek, amelyek modellvezérelt alkalmazásfejlesztést és szolgáltatásfejlesztési keretrendszereket használnak vállalati léptékű szolgáltatások és komponensek létrehozására, már legalább 60%-kal termelékenyebbek lehetnek, mint a hagyományos (3GL) integrált fejlesztési környezetek használói, és reagáló képességük is hasonlóképpen javulhat.

A szolgáltatásorientált alkalmazásfejlesztés módszereit (Service-Oriented Development of Applications – SODA) használva ma a szoftvercégek szolgáltatás-orientált üzleti alkalmazásokat (Service-Oriented Business Application – SOBA) fejlesztenek ki. Ez az alkalmazásokat lazán integrált (és jellemzően már kész) szoftverMODULokból (ún. szolgáltatásokból) építi fel, ellentétben a hagyományos szoftverfejlesztés „először programozni” (és csak azután összeépíteni) megközelítésével. A SOA-ra való áttérés folyamán a legnagyobb kihívást az üzleti folyamatok és adatok szemantikai egységesítése jelenti, amelynek révén ezen az új szinten kell megvalósulnia a folyamatok közötti gyakorlati együttműködésnek.

A SOA elterjedése a szoftvercsomagok licencvásárlása helyett a szolgáltatások előfizetése felé fogja eltolni a bevételeket, valamint a monolitikus szoftvercsaládoktól az összetett alkalmazások – azaz több, különböző szolgáltatásból összeépített alkalmazások – irányába való elmozdulást eredményez. 2008-ra az új alkalmazási szoftverekből származó bevételek nagy része SOA-t használó szoftvertermékekből realizálódik, akár hagyományos licenc-, akár előfizetési díjak formájában.



Emellett a szoftverintegrátorok és a szoftvergyártók közötti megkülönböztetés egyre inkább elmosódik, ahogy az alkalmazási csomagokat részekre bontják, és azokat SOBA-ként szállítják.

#### 2.7.4 Alkalmazásintegráció

Az alkalmazásintegráció a szoftveripar területén az innováció fő hajtóereje volt az elmúlt időszakban. A jövőbeni alkalmazásintegrációs termékeknél várható a központi, integrált metaadattár koncepciójának használata, ami az integrációval kapcsolatos összes metaadat egységes kezelését biztosítja.

Vállalati szolgáltatásbuszok (enterprise service bus – ESB) alakulnak ki, mint a köztes szoftverek új típusa, amelyek ötvözik a korábbi köztes szoftverekben található tulajdonságokat. Az ESB-ek szolgáltatásregisztrációt és -felkutatást kínálnak, a kommunikációban rugalmasságot, és a szolgáltatásminőség magas szintjét biztosítják. Az ESB-eket ki lehet majd terjeszteni teljes integrációs keretrendszerre is, üzletifolyamat-kezeléssel, B2B-képességekkel és alkalmazási adapterekkel bővítve.

Az alkalmazásintegráció egyéb, jelen pillanatban periférikusnak számító technológiái és platformjai pl. az ÁGENS-alapú integrációs eszközök, a szótár- (pontosabban: ontológia-) alapú transzformációs eszközök és az ún. VÁLLALATI INFORMÁCIÓINTEGRÁCIÓ (ENTERPRISE INFORMATION INTEGRATION – EII) is hatalmas lehetőségeket rejtnek magukban, és a szoftverinfrastruktúrák területén belül várhatóan itt történik majd a legtöbb innováció.

#### 2.7.5 Szoftverprojektek irányítása

A projektmenedzsment általános módszereinek és eszközeinek fejlődése keretében tovább terjednek a fejlesztést, mint csoportmunkát támogató eszközök (groupware), figyelembe véve a fejlesztő csapatok gyors átalakításának (rekonfigurálásának) igényeit is.

Emellett a szoftverfejlesztés esetében jelentős szerepet játszanak a különböző GYORSFEJLESZTÉSI MÓDSZEREK, amelyek gyakran terv-vezérelt (design-driven) megközelítésűek a fejlesztés gyors és iteratív megvalósítása érdekében. A GYORSFEJLESZTÉSI MÓDSZEREKEN belül az alábbiak segítségével a fejlesztés kockázatának radikális csökkentése érhető el:

- a követelmények fokozatos (iteratív) feltárása,
- az architektúra folyamatos hozzáigazítása a követelményekhez,



- a programozás jelentős részének kódgenerálással való kiváltása és
- a projektadminisztráció minimálisra csökkentése.

A szoftverfejlesztési projektek folyamatainak javításában egyre nagyobb szerep jut az empirikus kutatásoknak is, kísérleti módszerekkel (experimental software engineering) vizsgálják a módszerek és eszközök hatékonyságának alakulását különböző alkalmazási körülmények között.

### 2.7.6 Szoftverminőség

A teljes szoftverfejlesztési ráfordítás 40-50%-át a tesztelési, karbantartási költségek jelentik. Ezért várható olyan szoftvertechnológiai módszerek megjelenése, ill. elterjedése, amelyekkel ezek a költségek jelentősen csökkenthetők.

A gyakran több millió soros rendszerek folyamatos tesztelését segítik az ún. regressziós tesztelési eljárások, amelyek segítségével lehetőség nyílik arra, hogy csak a változások miatt módosult részeket és azok hatásait kelljen újra tesztelni. Ez a technológia a forrásprogramok nagyon pontos függőségi analízisén (szeletelés) alapul, amihez sok technikailag nehéz problémát kell megoldani, pl. hatékony statikus pointer-analízis.

A nagy szoftverrendszerek új (pl. web-es) környezetbe való telepítésénél (migration) sokat segíthetnek az olyan módszerek, amelyek segítségével automatikusan megállapíthatók a régi rendszer bizonyos funkcionálisai (aspect mining), illetve tervezési struktúrája (tervezési minták felismerése).

A szoftverek folyamatos karbantartása megköveteli a szoftverminőség folyamatos monitorozását is. Ehhez nyújtanak támogatást a szoftver metrikák, illetve az ezeken alapuló hibahajlandósági, karbantarthatósági, változtathatósági, megérthetőségi modellek, valamint a problémás program részek („bad-smells”) azonosítása és ezek (fél)automatikus transzformációja (refactoring).

### 2.7.7 Informatika mint közműszolgáltatás (IT Utility services)

Az üzleti folyamatokra, alkalmazásokra és infrastruktúrára vonatkozó követelményeket egyre inkább „erőforrásraktárak”-ból elégítik ki és nem egyedi, specializált erőforrásokkal. Az ilyen INFORMATIKAI KÖZMŰ (IT UTILITY – ITU) típusú szolgáltatás magában foglalja mind az infrastruktúra-hozszingot ill. infrastruktúraüzemeltetés jellegű szolgáltatásokat, mind az alkalmazás-hozszingot ill. alkalmazás-üzemeltetést.



A Gartner szerint 2010-re az alkalmazási igények 25%-a közműjelleggel biztosítottá válik, akár szervezeten belüli módszerrel, akár külső szolgáltatón keresztül (ez ma nem éri el az 5%-t). Az INFORMATIKAI KÖZMŰ a hardverkötségeket és a karbantartás munkaköltségeit a szoftver irányába fogja eltolni; a szoftverlicenckből származó bevételeket pedig a használat alapján történő fizetési konstrukciók felé. 2010-re a szoftverbevételek 30%-a nem licencekből származik majd, hanem „használat szerint” fizetett összegekből (ez ma nem éri el az 5%-t).