DR. VÁRKONYI ZSOLT

A szoftvermérnöki tevékenység* Magyarországon

Témakódok: 30 (szoftvergyártás)
84 (szoftvermérnöki tevékenység)

A szoftverfejlesztés magyarországi helyzetét vizsgálva, több olyan megállapítás fogalmazható meg, amely egybevág vagy nagyon hasonló a különböző országokban tapasztaltakhoz (pl. a szoftverfejlesztési eszközök használatának mértéke Magyarországon is nagyon alacsony). A hazai szoftvermérnöki tevékenység „state of the art" áttekintését megkísérlő dolgozat első része az ezen a téren meglevő magyar specialitásokra mutat rá. A második rész kiemeli a szoftverfejlesztési környezetek kialakításának fontosságát, rendszerezi a szoftvereszközöket, vizsgálja újrahasználhatóságuk problémakörét. A befejező rész a modern, előremutató szoftvermérnöki módszerekkel, eszközökkel foglalkozik (prototípus-előállítással, alkalmazásgenerátorokkal, negyedik generációs nyelvekkel, mikrogépek speciális szoftvereszközeivel), és kitekintést ad a szoftvermérnöki tudományt foglalkoztató kutatásokra.

(Érkezett: 1984. június 29.)

R. Yeh cikkében [1] megvizsgálta a szoftverfejlesztés helyzetét. Rámutatott a szoftveripar termelékenységét, a szoftver minőségét befolyásoló tényezőkre, vázolta azokat az új lehetőségeket, amelyek felé a szoftvermérnöknek nyitnia kell. Kiemelte a szoftverfejlesztési eszközök használatának fontosságát. Végül a szoftvermérnökkel szemben a következő elvárásokat fogalmazta meg: fejleszteni kell a programfejlesztési módszereket és környezeteket a termelékenység és a minőség javítása érdekében; ki kell lépni az eredményekkel a laboratóriumból, a módszereket és eszközöket a szoftveriparban kell meghonosítani.

R. Yeh amerikai–japán összehasonlításokkal érvelt. Felvetődhet a kérdés: mi a helyzet a mi környezetünkben? Miután megbízható adatok ezen a téren nem hozzáférhetők, egyéni tapasztalatokra támaszkodva kíséreljük meg áttekinteni a szoftverfejlesztéssel, a fejlesztési eszközökkel kapcsolatos helyzetet. Annak ellenére, hogy [1]-hez képest Magyarországról kicsit eltérő megállapítások fogalmazhatók meg a szoftvermérnöki tevékenységgel, ezen belül a fejlesztési módszerek és eszközök alkalmazásával vagy a szükséges kutatás-fejlesztésekkel kapcsolatban, megállapíthatjuk, hogy R. Yeh észrevételei a hazai szoftverfejlesztésekre is igazak.

Gazdasági, tudományos, oktatási környezetünkben a szoftvermérnöki tevékenység még csupán laboratóriumi szinten kapott létjogosultságot. A feladatok zöme ugyanis kicsi. Közepes vagy nagy feladat (>10 MFt, több mint 3–5 éves átfutású, több mint 5–10 főt foglalkoztató) elvétve akad. Ha lenne is, alig van merész megrendelő, és teljességgel hiányzik a megbízható fővállalkozó. Ezért azután a nagy feladatok szétesnek kisebb, apró, megoldható részekre, amelyekből – azt remélik – idővel integrálni lehet majd a kívánt komplex rendszert.

Környezetünknek a számítástechnikát illetően vannak még olyan sajátosságai is, amelyek nem közömbösek a szoftverfejlesztés fejlődése szempontjából. Ezek közül néhány így fogalmazható meg: még mindig alacsony szintű a különböző rangú felelős vezetők számítástechnikai tudása (többnyire felszínes, általános, a döntésekhez alkalmatlan). Uralkodik a rövid távú, közvetlenül, gyorsan megtérülő gazdasági szemlélet a szoftverfejlesztésben. Tervezéskor még azokban az esetekben sem gondolnak előre a szoftver karbantartására, ha azt többször kívánják értékesíteni. A szoftvereszközök fejlesztése – hacsak nem külön megrendelésre készül – nem fizetődik ki egy adott fejlesztés során, így még a nagyméretű, komplex rendszerek (információs rendszerek) létrehozásához sem készülnek a fejlesztést támogató eszközök. Az építőipari példák analógiájára ma még a fejlesztőknek inkább megéri a szoftverrendszer átadását követő hosszú, befejezhetetlen hibajavítási, módosítási periódus végigszenvedése, sokszor ingyen, mint az alapos rendszer-előkészítés, szoftvereszközökkel megtámogatott tervezés és kivitelezés, azaz a hosszú távra tekintő értelmes fejlesztés. Megéri, mert a nagy rendszerek megrendelői ma még nem versenyeztetik a vállalkozókat – igaz, egyik félből sincs sok –, a megrendelők tulajdonképpen ki vannak szolgáltatva a pár nagy fejlesztőintézet szolgáltatásainak. E fejlesztőintézetekben is lelassult a szoftverfejlesztéssel kapcsolatos kutatás, ugyanis a piaci kényszerhelyzet miatt az alig száz főre tehető valódi kutató-fejlesztő – a segéderőket, a programozó-kódolókat és vezetőiket nem számolva ide – java része azonnali fogyasztásra ír programot. Valójában csak egy-két, egyelőre laborszintű fejlesztésről beszélhetünk ezen a téren (pl. SOFTING, ISAC–MOZART, QUALIGRAPH...), és a velük kapcsolatos tapasztalatok is inkább külföldi alkalmazásokból szerezhetők be.

*Software engineering.

Problémák származhatnak a szoftverfejlesztők képzettségéből is, ugyanis annak ellenére, hogy a hazai fejlesztők kvalifikációja magas – az egyetemet végzettek aránya általában nagyobb a nyugati országokban alkalmazott fejlesztőkénél –, a szoftverfejlesztés kutatása nem hódított kellő tért egyetemeinken. A számítástudománynak például nincs egyik egyetemünkön sem önálló kara, intézete stb. Igaz, ritka az olyan szakember, aki gyakorlati tapasztalatai alapján bátran taníthatná ennek egyes fejezeteit.

Nagyon sok múlik azon, hogy egy-két fejlesztőhelyen van-e kezdeményező, kiemelkedő képességű, a szoftverfejlesztés technológiája iránt vonzódó és vezetésre is képes szoftverfejlesztő. Ha van, és a mikrokörnyezete nem gátolja működését, akkor – elsősorban az irodalomból származó információkra támaszkodva – lehetősége nyílik a modern szoftverfejlesztés módszereinek, eszközeinek „újra-megalkotására" (ez a hazai gyakorlat), és az így szerzett tapasztalatok alapján „tudása gyarapítására". Nehézségek abból adódhatnak – ebben a megközelítésben –, hogy az irodalomban található módszer, eszköz mögött fejlett számítástechnikai környezet áll (amiről nem szokás írni, ugyanis ez ott természetes), és a hazai adaptáláskor, az újra-megvalósításkor általában el szokták felejteni ezt a tényt. Pedig a sok kudarc visszavezethető lenne arra, hogy nem vették figyelembe az adaptálás során a két környezet különbözőségeiből adódó, alig kezelhető problémákat!

Amíg az IBM, Honeywell, CDC, HP, DEC, ICL, Siemens stb. gépekre készült szoftver fejlesztéséről szóló beszámolókban keveset olvashatunk a hardvermegbízhatóság szempontjának kiemelkedő fontosságáról, addig a hazai géppark java részét kitevő közepes (nagy) és mini kategóriákra épülő szoftver fejlesztőinek ezzel a kérdéssel komolyan kell foglalkozniuk. Nagyobb állomány- és logikai komplexitású rendszereket fejlesztő szoftvermérnökökben többször felvetődött e hardverállapotból eredő megbízhatósági probléma alkalmazott kutatásának szükségessége. A téma ma még – figyelembe véve a háttértár-forrásainkat és lehetőségeinket – nem tekinthető túlhaladottnak.

Megkockáztatható a következő megállapítás is: tapasztalataink szerint, ha korszerű és megbízható gép mellett feltűnik egy kreatív, érdeklődő, viszonylag kiművelt fő, akkor tekintet nélkül előképzettségére, a modern, megbízható gép környezete kikényszeríti belőle az értelmes módszer-, eszközhasználatot és az elvárható minőséget a szoftverfejlesztésben. Úgy tűnik tehát, a számítógép meghatározza a szoftverfejlesztés színvonalát, minőségét.

Szoftverfejlesztéseink (nem az egyszerű programírásra gondolunk!) mindig valamilyen köntösbe csomagolva kerülnek a köztudatba. Ennek valószínű oka az, hogy a fejlődést elősegítő, támogató központi pénzalapok megszerzéséhez szükség van az aktuális – országosan éppen divatos – célokra orientáltság kinyilatkoztatására. Most még a „hálózatok" kialakítása van napirenden. Ennek hatására is erőltetjük speciális hardver kiépítését, TAF-vezérlő szoftver készítését; viszonylag sok cikket jelentetünk meg ebben a témában, ugyanakkor tudjuk, hogy bizonyos, lokálisnak nevezhető távolságot (vagy ezen belül bizonyos felhasználói számot) túllépve, beleütközünk infrastruktúránk egyik nagy problémájába, a megbízhatatlan távolsági információtovábbításba. Szerencsére egyre többen vannak azok, akik élő, használható rendszerük létrehozása során felismerték ezt a nehézséget, és megoldásként az osztott, helyi igényeket kielégítő információfeldolgozást kezdik szorgalmazni. Ez pedig újabb, speciális szoftverfejlesztéseket, szoftvermérnöki kutatásokat kíván. Emellett a professzionális mikroszámítógépek gyors fejlődése is egészen új környezetet kínál a szoftverfejlesztőknek az eddig nehezen elképzelhető feladatok megoldására.

Áttekintésnek szánt dolgozatunkban nem törekedhettünk teljességre, alapos elemzésekre, ezért a szoftverfejlesztési módszerekre, eszközökre összpontosítottunk. Hivatkozásainkkal, példáinkkal a téma iránt érdeklődőknek kívánunk segítséget nyújtani.

A szoftvermérnöki feladatok között eligazodni nem nehéz. Adott esetben segítségül hívhatók a különböző osztályozó, kategorizáló dolgozatok. Amíg az egyik dolgozat a szoftverfejlesztés életciklusának fázisaihoz illeszti a szoftvermérnöki tevékenységeket, addig a másik a szoftvermérnöki feladatokat típusokba sorolva osztályozza. Vannak olyan munkák, amelyekben csak a szoftverfejlesztési eszközöket tekintik a szerzők a szoftvermérnöki területhez tartozónak, vannak olyanok is – és ezek az előremutatók –, amelyek szoftverfejlesztési típusokat különböztetnek meg, ezekhez pedig különböző fejlettségű ún. „szoftverfejlesztési környezetek"-et határoznak meg. E környezetekben azután megkülönböztetik a fejlesztés különböző fázisaihoz tartozó fejlesztési eszközöket. Meg kell jegyeznünk, hogy a fejlesztési eszközök kifejezés használata sem egyértelmű az irodalomban: sokan csak programeszközöket értenek rajta, mások pedig a módszereket, elveket, standardokat, segédleteket, könyvtárakat, azaz a nem programeszközöket is ide sorolják.. Ezekről adnak áttekintést a [2], [3], [4] és [5] munkák. Szoftverfejlesztési eszközök szűkebb értelemben – az NBS* definícióját is felhasználva [2] – mindazok a számítógépi programok, amelyek támogatják a fejlesztés különböző fázisaiban folyó munkát, tehát a specifikálást, a tervezést, a tesztelést, a bevizsgálást, a dokumentálást, a karbantartást és a fejlesztési munka irányítását. Közismert eszközök tartoznak ide, például a fordítóprogramok, szövegszerkesztők stb. Az [5] fogalmazásában vannak olyan alapvető eszközök, amelyek minden fejlesztési környezet részei, az előbbieken túl ilyen például a szerkesztőprogram (linkage editor), a futási idő alatti segédprogramok, a forráskódban hibát kereső programok. Ide kell sorolni a még általánosan el nem terjedt, de már használatba vett eszközöket: a tervezést támogató programokat, programanalízist végző programokat és a tesztelési programokat, sőt azokat is, amelyekről már beszerezhetők ugyan információk, de maguk az eszközök többnyire még laboratóriumszintű kutatási állapotban vannak (például a programot formálisan ellenőrző rendszerek vagy a komplex programfejlesztési környezetek). Tágabb értelemben a szoftverfejlesztési eszközökhöz sorolhatjuk a nem programeszközöket is, mint a különböző diagramokat (adatáramlás, HIPO, Chapin-chart), a módszereket (Jackson-), a szabványokat, a fejlesztés vezetését, infor-

* National Bureau of Standards = az Egyesült Államok Szabványügyi Hivatala.

málását támogató kimutatásokat is. A kereskedelemben kapható eszközök száma ezer körül mozog. A [6]-ban hirdetett Automated Tools Index 750 db automatizált szoftvermérnöki eszköz részletes leírását kínálja.

A hazai használatról alkotható kép, ha a *Számítástechnika* utolsó harminc száma – 1981/11-től 1984/4-ig – alapján kívánnánk tájékozódni, lehangoló lenne. Ha az OSAK-reklámokat és az új géptípusok ismertetése során említett operációs rendszerek verzióiról szóló tájékoztatókat (4–5 db ebben az időszakban) nem számítjuk, akkor nem jelent meg 10 cikk sem összesen ebben a témában. Ezek között is a reklámízű tájékoztatás szintjén marad az ANSWER/CDL2 nyelvi rendszerről, a MOZ-ART-ról, a SIDES-ról szóló, többet nyújt az MPROLOG és a GESAL bemutatása. Igazán felhasználható információkat csak az éles működtetés alapján szerzett tapasztalatokból származó beszámolók adtak, mint például a BUDAPRINT-nél, a KSH Számítóközpontjában és az ÉGSZI-ben folyó nagyüzemi szoftvergyártásról szólók. Valószínűleg nem ennyire rossz a helyzet az eszközök alkalmazása terén. Tudomásunk van többek között arról, hogy a SZÁMALK-ban szoftvertervezést támogató eszközök fejlesztése folyik, vagy az SZKI-ban kifejlesztett QUALIGRAPH a szoftveranalízisnek, -bevizsgálásnak hatékony eszköze stb., illetve a hazai HP 3000-es környezetben jók a tapasztalatok a skeleton (váz-) programozás, a magas szintű, párbeszédes kódgenerátorok használata terén. Az utóbbiak ma a legmodernebb eszközök közé sorolhatók. Emlékszünk azonban arra is, hogy a 70-es évek végén létezett itthon egy COBOL nyelvű fejlesztést támogató (akkor modern) eszköz, a SERIES–J, amely mégsem terjedt el Magyarországon – vélhetően igényhiány miatt. Azon sem lehet csodálkozni az előzőekben vázolt „sajátosságok" alapján, hogy a 80-es évek elején indított, központi finanszírozású, szoftverfejlesztést támogató szoftvereszközök és módszerek (hazai gyűjtőnevén: szoftvertechnológia) létrehozását kitűző program munkakoncepciójában megfogalmazott „prototípusrendszer-készítés" feladatköre nem talált megértésre. Pedig a nagy teljesítményű és viszonylag olcsó professzionális mikroszámítógépekkel elérhető, hogy a felhasználó kiszolgálásához vagy meggyőzéséhez ne 100 oldalnyi gépelt rendszertervdokumentációra legyen szükség, hanem a közvetlen, kézzelfogható tapasztalat, kidolgozott mintarendszer alapján döntsön. Sőt az ehhez szükséges kiszolgálószoftver is elérhető már, például egy relációs adatbázist kezelő szoftver, a CONDOR, mikrogépekre.

A szoftver teljes életciklusának nyomon követése nálunk még nem létszükséglet. Általában annyira elhúzódik a nagyobb rendszerek fejlesztése, hogy nem lehet az életciklus egyes fázisait tisztán megkülönböztetni. Valójában az első „kísérleti" átadást követően folyamatos további fejlesztés tapasztalható, ami sohasem ér el a klaszszikus értelemben vett karbantartási stádiumba. A fejlesztés ugyanis gyakorlatilag két nagy szakaszból áll: „előfejlesztés"-ből, ami a (kísérleti) átadással zárul, ehhez társul a költségek majdnem 100%-a; ezt követi a „továbbfejlesztés" egészen addig, amíg ez a folyamat – új gép beérkezése, új rendszer elkészítése, érdektelenség stb. miatt – meg nem szakad.

Ha ez a megállapítás elfogadható, akkor kézenfekvő lenne az előfejlesztést prototípus készítésének tekinteni – természetesen ekkor valóban *prototípust* kellene készíteni is! –, és a továbbfejlesztés volna a tényleges fejlesztési időszak. A két szakasz költségeinek a 40%–60% arányú megoszlása lenne reális. Valószínű, hogy akkor kevesebb szerződésteljesítési, feladatmegfogalmazási, -megoldási problémával kellene szembenézniük a fejlesztőknek és megbízóiknak hosszú távon.

A szoftvereszközök használatát tanítani lehetne – és kellene is. Javulás várható ebben a tekintetben is a professzionális mikrogépek elterjedésétől.

Az országban igen nagy mennyiségű, zömében egyedi igényt kielégítő program, programmodul készül. Ezt a szellemi terméket, néhány esettől eltekintve, általában nem használják fel újra. Az okok között megtalálható egyebek mellett a terjesztésre való felkészítés (kiszerelés) igénytelensége, a sokszor teljességgel hiányzó dokumentáció; és az is, hogy hiányzik az országosan elfogadott nyilvántartásba vétel végrehajtása, a megfelelő könyvtár; nincsenek a különböző forrásnyelvű szoftver konverzióját biztosító eszközök, sem alkalmas és érdekelt szervezet stb. Hiányzanak tehát azok az általános érvényű elvek, amelyek alapján a fejlesztő konkrét programját, modulját, szubrutinját eleve úgy készítené el, hogy az mások számára felhasználható legyen. Ezen a téren valódi eredmény eddig csak egyéni kezdeményezésekből, kis méretben, helyi jelleggel született. Az OSAK vagy más cégek hasonló jellegű tevékenységében nem tapasztalható még ilyen igényesség. Célszerű lenne és a népgazdaság szellemi potenciáljával való értelmesebb gazdálkodás felé mutatna, ha egy alkalmas vállalat ezt a munkát felelősséggel elvállalná. Gondoskodni kellene a különböző szempontok szerinti osztályozási rendszerekről (a besoroláshoz, a visszakereséshez, vö. [2], [3]), alkalmas adatbázis-kezelőről, amely a szoftver alkotórészeit és tulajdonságait tárolná, katalógusokról a visszakereséshez, információs eszközökről a tájékoztatáshoz, egységes, számítógépben tárolt dokumentációkról, amelyek a programmal együtt könnyen karbantarthatók, specifikációs szabványokról stb. A feladatot országosan egy kereskedelmi cég – bizományosi konstrukcióban – is elláthatná. Szüksége lenne azonban igen komoly erőt képviselő szoftveres szakértőkre, szabványosításra, alkalmas, nagy *kiépítettségű* gépre (nem feltétlenül nagygépre!). Úgy látszik, ez a ma még nem hasznosított erőforrás megfelelő szervezet kezében talán a legelsőként mobilizálható, vonható a termelésbe, ami a szoftvermérnöki munka fejlődésére, a szoftverfejlesztési eszközök készítésére is nagy hatással lehet.

Sajátosan hazai vonás, hogy 1976-tól kezdve a szoftverfejlesztők nagy része jelentős központi támogatással szoftverfejlesztési környezetek kialakításán munkálkodik. Irodalmi adatok, kis tanulmányutak felcsillantották a reményt, hogy a CADES-hez (az ICL New Range szoftverfejlesztési környezete) hasonló, ám modernebb, integrált eszköztárat fogunk készíteni. Természetesen ekkor már az *alkalmazási* programrendszerek fejlesztésének támogatása volt a cél. Az erre vonatkozó kutatások – a hazai számítógép-környezet, a fejlesztési nehézségek és az idők során átformálódott igények figyelembevételével – elvezettek az ANSWER *rendszer*program-fejlesztési környezet megalkotásához, amely először a CDL rendszerprogramozási nyelven egy UNIX-szerű operációsrend-

szer-magon, majd a CDL fejlettebb változatának, a CDL2-nek a környezetébe integrált rendszerprogramon alapul. Ennek a részeit már pár év óta hatékonyan használják az Ada rendszerprogramozási nyelv kivitelezéséhez. (Hatékonyan tudják használni, mert hatékony a *nagyszámítógép,* amelyen a fejlesztés folyik!) Azóta már napirenden van az Ada szoftverfejlesztési környezet létrehozása is, szintén hazai fejlesztésben.

Az eredeti cél torzulását viszonylag hamar felismerték, és 1978-tól intenzív kutatás kezdődött alkalmazásfejlesztési környezetek megalkotására. Az akkori irodalom a PROTEE-t, a HIPO-t stb. jelölte meg az éppen bevezetett divatos megjelölés szerint „technológiá"-nak, a fejlesztőknek pedig rövid idő alatt rá kellett jönniük arra, hogy egységes, a szoftver teljes életciklusát lefedő, eszközökkel is támogatott „technológia" nincs. Ez a felismerés vezetett el a részekből, mozaikokból összerakható elvek, ajánlások, módszerek és eszközök együtteséhez, a MOZ-ART-hoz, és ehhez az áramlathoz kapcsolódtak a külföldi együttműködés alapján fejlesztés alatt álló SOFTING, a hazai SOMIKA, valamint az első hazai szoftverminősítési kísérletek. Egyik sem volt a teljes életciklust végigkísérő technológia – a SOMIKA nem is azt a célt szolgálta, ugyanis ez a kész szoftvert bevizsgáló eszköz –, de mind a MOZ-ART, mind a SOFTING tervezői törekedtek a programtervezési, kódolási, programbelövési fázisokon kívüli életszakaszok kezelésére is.
Amíg az eddig felsorolt környezetfejlesztési kísérleteink részben irodalmi, részben külföldi kooperációs hatásra, valójában esetleges módon fejlődtek, és e környezetek meghatározó nyelve az ANSWER CDL2-jét kivéve egyikét elterjedt, közismert, úgynevezett konvencionális vagy harmadik generációs nyelv, addig – szintén sajátos helyzetünkből származtathatóan – a nem procedurális nyelvek fejlesztése terén nemzetközi mértékkel is sikeres kísérleteket folytattunk a PROLOG-gal. A PROLOG nyelvre támaszkodó MPROLOG környezet pedig túlmutat a negyedik generációs szoftveren az ötödik generáció felé (amint Japán ötödik generációs programja is bizonyítja).

Úgy tűnik, elég sok szoftverfejlesztési környezet fejlesztése folyt már ahhoz, hogy a különböző fejlesztéseket valóban összehangolják, és az sem ártana, ha az ilyen irányú kutatásokat a felhasználói területek szükségletei szerint osztályoznánk és mérlegelnénk. Az automatizált szoftverfejlesztési környezetek létrehozása helyes útnak tekinthető. Ezt hangsúlyozza [1] és [5], erre figyelmeztet Japán, erre érzett rá Angliában a japán ötödik generációs projekt hatására létrejött „Alvey software engineering" stratégia (vagy Alvey-program) néven emlegetett válasz-projekt.

Európa más vezető szoftveresei is felismerték – már jóval korábban – a szoftverfejlesztési környezetek készítésének és alkalmazásának szükségességét ahhoz, hogy használható minőségű szoftvert tudjanak előállítani elfogadható időn belül és elfogadható áron. Közülük az egyik a svéd SYSLAB, amelynek a neve a laboratóriumi kutatásokra utal; a másik, az NCC és a nyugatnémet GMD közös SDSS (Specification and Development of Software Systems) projektje pedig a szoftverfejlesztés teljes menetét kívánja lefedni.

A svéd fejlesztők már 1980-ban, a SYSLAB egyes részeinek kidolgozásában mikroszámítógépet alkalmaztak. Valójában a mikroszámítógépek mára már elérték, sőt meghaladták az ismert közép- és nagygépek teljesítményeit; a RAIR 32 bites szuper-mikrogép például a legnagyobb VAX gépet is túlszárnyalja, ugyanakkor a nagy kapacitású Winchester-tárolók is piacra kerültek. Ezeknek a berendezéseknek – a kényelmen túl – a viszonylagos olcsóságával új fejezet és lehetőség nyílik a szoftvermérnöki tevékenység előtt. Példaként említhető a gyorsan terjedő ún. „windowing" (láttatás, ablak) technika (pl. a VisiON), a „képi szoftver" (pl. VisiCalc; 1–2–3) stb.
Nem hagyhatjuk figyelmen kívül, hogy a döntés-előkészítő rendszerek, az egységes hivatali rendszerek stb., mint a mikrogépek egyre inkább előtérbe kerülő alkalmazásai, új megközelítésű szoftverfejlesztést igényelnek. Olyan szoftver-eszközkészlet kifejlesztéséről van szó, amely lehetővé teszi a gyors rendszer-összeépítést, -kiterjesztést, -módosítást. Ezek az eszközök egyre differenciáltabbak, egyre speciálisabbak lesznek. Az említett képi szoftvereken (iconic software) és az ablaktechnikán túl természetesen szükségesek még az úgynevezett analitikus szoftverek is, például a gazdasági tervezéshez, a modellezéshez alkalmas nyelvek, az adatkezelést biztosító szoftver.
Új fejezet az „alkalmazásgenerátorok" megjelenése, a nem procedurális felhasználói nyelvek igénye. Az alkalmazók kényelmének a kiszolgálása volt többek között az a hajtóerő, amely elvezetett a szoftver fejlődésében az összefoglaló néven „negyedik generációs nyelvek"-nek nevezett nyelvek kialakulásához. Röviden úgy lehetne jellemezni őket, hogy használatukat két nap alatt elsajátíthatja akár az alkalmazó, akár a szoftverfejlesztő, és segítségükkel egy nagyságrenddel gyorsabban lehet alkalmazási rendszereket kifejleszteni, mint mondjuk COBOL vagy PL/I nyelven. E nyelvek szintaxisa éppen ezért problémaorientált. A többnyire nem procedurális szerkezetű nyelvek lehetőséget adnak a hagyományos, szigorúan strukturált megközelítés helyett a tapasztalatok szerint gyorsabb, megbízhatóbb és közvetlenül, könnyebben elfogadható végeredmény előállítására azáltal, hogy képesek a „prototípus megközelítésé"-re. Ennek az a lényege, hogy a nagyméretű specifikációk helyett prototípus-funkciókat fejlesztenek ki, amelyek lehetővé teszik, az első részeredmények megjelenésétől kezdve, a felhasználó észrevételeinek és kívánságainak megvalósítását. (Ígéretesek a MAPPER, Mantis, LINC, RPG III, PRO–IV nyelvekkel kapcsolatos tapasztalatok és eredmények.)

Annak ellenére, hogy Hoare egyik ez évi előadásában a magas szintű specifikációs nyelvekben jelölte meg a jövő fejlesztési eszközeit az előretörő logikai, nem procedurális nyelvekkel szemben, nem lehet elvitatni az utóbbiak létjogosultságát és a bennük rejlő lehetőségeket. Ezt ismerték fel a japánok akkor, amikor a PROLOG-ot választották ötödik generációs programjuk központi nyelvévé, ezt igazolják a mesterséges intelligencia kutatásának témakörébe tartozó úgynevezett tudásreprezentációs nyelvek, szakértői rendszerek választott nyelvei, amelyek között a PROLOG-nak szintén meghatározó szerepe van. Sajátos körülményeink – szerencsére – lehetővé teszik

azt is, hogy a harmadik generációs számítástechnikai környezetünkben az előbbiekben felsorolt, az ötödik generációs nyelvek felé mutató kutatásokat, fejlesztéseket is folytassunk. Ezek pedig a jövőben a szoftvermérnöki tevékenység ígéretes fejezetei.

Irodalom

[1] YEH, R.: Software engineering. (In: Tomorrow's Computers; The Challenges) = IEEE Spectrum, Nov. 1983. pp. 91–94.
[2] Software Development Tools. (prep. by the U.S. National Bureau of Standards) = Software World, Vol. 12. No. 4.
[3] The New [1982] Computing Reviews Classification System, Final Version. = CACM, Vol. 25. No. 1.
[4] REIFER, D. J.–TRATTNER, S.: A Glossary of Software Tools and Techniques. = Computer, July 1977.
[5] HOWDEN, W. E.: Contemporary Software Development Environments. = CACM, Vol. 25. No. 5.
[6] Testing Techniques Newsletter (Systems Research Association). Vol. 6. No. 2.

Summary

While examining the Hungarian software development situation, several such findings may be formulated which correspond or are very similar to those of other countries (e. g. the rate of using software development tools is very low in Hungary too). The first part of this overview of the software engineering state of the art in Hungary tries to point out the Hungarian specialities in this field. The second part emphasizes the importance of software development environments, systematizes the software tools and examines the problem of their repeated utilization. The concluding part deals with the up-to-date software engineering methods and tools (prototype design, application generators, fourth generation languages, special software means of microcomputers) and gives an outlook to research efforts carried out in the field of software engineering sciences.

Резюме

Исследуя положение развития программного обеспечения в Венгрии, можно формулировать несколько установлений, которые совпадают или подобны к испытанным в разных странах (напр. уровень применения средств развития программного обеспечения очень низкий в Венгрии).
Первая часть работы попытающей осмотреть положение венгерского инженерства по программному обеспечению указывает на венгерские особенности этой области.
Вторая часть подчеркивает важность формирования среды развития программного обеспечения, систематизирует средств программного обеспечения и исследует проблемы их повторного использования.
Последняя часть занимается современными прогрессивными методами и средствами инженерства по программному обеспечению (созданием образцов, генераторами применения, языками четвертого поколения, специальными средствами программного обеспечения микро-ЭВМ), и также рассматривает исследования в области инженерной науки программного обеспечения.

A robotok előretörése Nyugat-Európában

Az újraprogramozható robotok száma az elmúlt években gyorsabban emelkedett Nyugat-Európában, mint akár Japánban vagy az Egyesült Államokban – a British Robot Association februári felmérése szerint. 1983-ban 12 500 robot volt Nyugat-Európában, ez 37%-os növekedés. Japánban 27% (16 500 db), az USA-ban pedig 28% (8000 db).
Az európai rangsorban Olaszország feljött a harmadik helyre (700-ról 1800 db), az NSZK maradt az élen (37%-os növekedés, 4800 robot), Svédország a második (24%, 1900 db) és Anglia foglalta el a negyedik helyet 1753 robottal az 1500 robotot alkalmazó Franciaország előtt.
Az angol robotgyártók növelték eladásaikat a hazai piacon (23%-ról 34%-ra), és ugyancsak növelte angol exportját az USA – más európai és japán szállítók rovására. Angliában jelenleg 58 cég foglalkozik robotok forgalmazásával, ezek mintegy fele hazai gyártmányokat kínál. A forgalomhoz képest ez túl sok, így a piac átrendeződését várják a BRA szakértői.

American Machinist, 1984/4. szám

Május 13. és 29. között rendezték meg a párizsi kiállítási parkban a 13. Szerszámgép-, hegesztési és gépi berendezés biennálét, amelyen 23 ország 1400 vállalata képviseltette magát. Az NC gépek és a programozható automaták újabb típusai autodiagnosztikai eszközt is tartalmaznak. A fejlesztés iránya az automatizált gépcsoportok és a számítógépes termelésvezérlési és tervezési rendszerek integrálása felé mutat.

01 Hebdo, 796. szám

RAM (Robots Autonomes Multiservices) néven nemzetközi robotikai program indult, francia és japán vezetéssel. A program kiterjed az űrkutatás, a mélytengeri kutatások, az atomtechnika, a bányászat, a mezőgazdaság, az orvostudomány, a közmunkák, az ipari karbantartás, a kármegelőzés és a közszolgáltatások területére. Japánon és Franciaországon kívül részt vesz benne az USA, Kanada, Anglia, az NSZK és Olaszország is. Anglia, Franciaország és az NSZK már 1979-től szorosan együttműködik a robotika fejlesztésében. A franciák 1984-ben 40 millió frankot fordítanak a program fedezésére.

01 Hebdo, 799. szám

A Siemens cég az Ethernet hálózatot használja fel NC-gépeinek és automatáinak összekapcsolására. A választást az Ethernet előnyös jellemzőin kívül az is motiválta, hogy a cég korábban már irodai hálózatában is ezt alkalmazta. Az üzemi automatizálásban ez a megoldás egy 1 X 1,5–2,5 km területű helyi hálózatot tesz lehetővé.

01 Hebdo, 801. szám

Június 5. és 8. között tartották Párizsban a Nemzetközi Informatikai és Automatizálási Napok (JIIA) első rendezvénysorozatát *Usinica* címmel. A programon hét téma szerepelt: a valós idejű termelésvezérlő rendszerek; a helyi hálózatok és műhelyterminálok; a minőségellenőrzés; a CAD/CAM rendszerek; az automatizálás és végül a robotika. Az első két napon nyilvánosan teszteltek hét termelésvezérlő rendszert.

01 Hebdo, 801. szám

Az ötödik generációs számítógép fejlesztésében Sódzsí Tanaka professzor szerint 100 Mbájt kapacitású lapkákra van szükség. Szerinte ez még megoldható szilícium alapon, de 0,7 mikrométer finomságú megmunkálást kell elérni hozzá. Egyetlen ekkora kapacitású lapkára harmincpercnyi szöveget lehet majd rögzíteni, ami lehetővé teszi gazdaságos beszélőgépek kifejlesztését.

01 Hebdo, 801. szám