Biztonságos és megbízható számítástechnika

A vendégszerkesztő bevezetője

selenyi@mit.bme.hu

Közismertek olyan alkalmazási területek, ahol a rendszer hibás működése jelentős anyagi kárt tud okozni vagy emberi életeket veszélyeztet. Ilyen kiemelten kritikus alkalmazások, például a

– közúti forgalomvezérlés, vasútbiztosítás,
– nagyméretű adatbázisok szervezése, működtetése,
– helyfoglaló rendszerek,
– az intelligens gépkocsi,
– az űrkutatás stb.

A *nagymegbízhatóságú rendszerek* fogalomköre az elmúlt néhány évtized alatt a fenti alkalmazási területeken kidolgozott megbízhatóság javító megoldások együtteséből alakult ki, és összefogja mindazokat a módszereket, alkatrészeket és technológiákat, amikkel a *mindenkori átlagos alkalmazásokhoz* képest megnövelt megbízhatóságot lehet biztosítani.

A címben szereplő két rokon értelmű fogalom – biztonságos, illetve megbízható – a rendszer helyes, vagy helytelen működésének két különböző aspektusára koncentrál. Egy rendszer akkor *megbízható,* ha nagy valószínűséggel helyesen működik, és akkor *biztonságos,* ha elromlás esetén sem okoz katasztrofálisan nagy bajt. Például a liftek (általában) megbízhatatlanok, mert sokszor elromlanak, de biztonságosak mert hiba esetén (általában) nem okoznak katasztrófát. Mindkét fajta biztonsági tulajdonság tervezhető, ehhez nyújtanak alapot a nagymegbízhatóságú rendszerek..

A megbízhatóság növelésének alapvető két útja:

– A *nyers erő* módszere, amikor különlegesen megbízható alkatrészekkel és technológiákkal dolgozunk.
– A *rendszertechnikai* módszer, amikor adott tulajdonságú alkatrészek speciális összekapcsolásával – tehát rendszertechnikai úton – javítjuk a teljes rendszer megbízhatóságát. Ebben az esetben tehát a rendszer jobb, mint az alkatrészei. A megbízhatóság növelésének rendszertechnikai útját nevezik hibatűrő megoldásnak (angolul: fault tolerant system, fault tolerant computing).

A nyers erővel tehát a hiba keletkezésének valószínűségét kívánjuk csökkenteni, míg a hibatűréssel azt próbáljuk elkerülni, hogy a rendszer belsejében keletkező hiba(ok) kijusson a rendszer kimenetére, azaz a felhasználói szinten hiba(jelenség) legyen.

A mai informatikai-számítástechnikai világban a megbízhatóság két pillére – a megbízható alkatrész és a hibatűrő rendszertechnika – mindig együtt van, de időről időre fontosságuk, fejlődésük ritmusa változik, a szakma az idők során hol az egyik, hol a másik oldalt hangsúlyozza és fejleszti. Jelenleg a *hibatűrés* hangsúlyozása látszik erősebbnek, azaz annak a folyamatnak vagyunk vagy leszünk tanúi, amikor a hibatűrő rendszertechnika, a különböző hibatűrő megoldások már nem csak e bevezetőben említett különleges alkalmazásokban jelennek meg, hanem bevonulnak a hétköznapi alkalmazásokba is.

Arról van ugyanis szó, hogy a hétköznapi informatikai-számítástechnikai rendszerek is (belülről nézve) egyre komplexebbek lesznek és ezek a nagy rendszerek elviselhető költségű átlagos alkatrészekből felépítve csak akkor tudnak már működni, ha például hibavédő kódokat, belső ellenőrzéseket, tartalék egységeket stb. – összefoglalóan hibatűrő megoldásokat – tartalmaznak.

A hibatűrő rendszerekre vonatkozó ismeretek ma és a közeljövőben már nem csak a különleges alkalmazásokkal foglalkozó kevesek szükséges ismeretei, hanem az informatikai-számítástechnikai világ általános metodikájának részei.

A nemzetközi szakmai életben az utóbbi tíz évben jelentős hangsúlyt kaptak a biztonságos, a szolgáltatásokat megbízható módon nyújtó számítástechnikai rendszerek felépítésének általános kérdései. A *Dependable Computing* elnevezéssel összefogott szakterületnek egyre több rangos konferenciája van és 2004-ben az IEEE is elindította *Dependable and Secure Computing* című folyóirat-sorozatát.

A hazai tudományos életben is korán felismertük a szolgáltatásbiztos számítástechnika fontosságát. A hazai eredmények nemzetközi elismertségét bizonyítja, hogy idén áprilisban a Budapesti Műszaki és Gazdaságtudományi Egyetem Méréstechnika és Információs Rendszerek Tanszéke rendezi meg az ötödik európai szakkonferenciát, a Fifth European Dependable Computing Conferrence-t. Az e számunk megjelenése előtt, – április 20-22. között – megtartott konferencia részletes anyaga megtalálható a **http://sauron.inf.mit.bme.hu/EDCC5.nsf** webcímen.

A Híradástechnika folyóirat e havi számában főként az informatikai rendszerek megbízhatóságához kapcsolódó cikkek szerepelnek. A válogatások a hazai szolgáltatásbiztos számítástechnika művelői közül két nagy műhelyre, a BME Méréstechnika és Információs Rendszerek Tanszékre, valamint a Széchenyi István Egyetem Informatika Tanszékre alapoznak.

Dr. Selényi Endre

Szoftverrendszerek tesztelési modellje

DR. SZIRAY JÓZSEF

Széchenyi István Egyetem
sziray@sze.hu

Reviewed

Kulcsszavak: szoftvertesztelés, verifikáció, validáció, hibamodellek, formális módszerek

A cikk komplex szoftverrendszerek tesztelésének általános szempontjaival foglalkozik, a hangsúlyt a biztonságkritikus számítógéprendszerek szoftverjére helyezve. Először a számításba veendő szoftver hibákat ismerteti, majd ehhez kapcsolódóan a verifikáció és validáció feladatait. Ezt követően egy általános leképezési séma kerül ismertetésre, amely egy adott szoftver bemeneti és kimeneti tartománya közötti egy-egy értelmű kapcsolat leírására szolgál. Erre a sémára egy tesztmodell épül, amely tartalmazza a különböző hibaosztályokhoz tartozó tesztinputokat.

A közölt tesztmodell magában foglalja mind a verifikációs, mind pedig a validációs folyamatokat. A modell jelentősége abban áll, hogy megkönnyíti a világos megkülönböztetést a verifikációs és validációs tesztek között. Mindez a tesztek megtervezésének és kiértékelésének folyamatában bizonyul fontosnak és hasznosnak, mivel a két tesztelési feladat egymástól lényegesen eltérő megközelítést tesz szükségessé.

A cikk befejező része azt az esetet vizsgálja, amikor a szoftvert formális módszerek alkalmazásával tervezték meg. Ebben a részben tárgyalásra kerülnek a formális módszerek alkalmazásának következményei és problémái, valamint a módszereknek a verifikációra és a validációra való hatása.

1. Bevezetés

Ismeretes, hogy a komplex szoftverrendszerek megbízható üzemeltetése, felhasználása szigorú követelményeket támaszt a fejlesztésükre vonatkozóan. Ebbe szorosan beletartozik az a minőségbiztosítási technológia, amely a teljes fejlesztési folyamatot végigköveti, a specifikáció megadásától a kész rendszer üzembe helyezéséig [1-3]. A minőségi és megbízhatósági követelmények kielégítése szükségessé teszi azt, hogy a szoftvert úgynevezett *verifikációs* és *validációs* eljárásoknak vessük alá [1-7]. A verifikációban az egyes fejlesztési fázisok közötti összhang ellenőrzése a feladat, míg a validációval a végső rendszer ellenőrzésére kerül sor, annak eldöntésére, hogy az mennyire felel meg a felhasználó által előírt követelményeknek.

A verifikálási-validálási tevékenység pontos végrehajtása a következő főbb előnyökkel jár:

- Segít annak eldöntésében, hogy elkezdhetjük-e a fejlesztés soron következő fázisát.
- A fejlesztési folyamat korai szakaszában mutathat ki problémákat, hibákat.
- A szoftver minőségére, megbízhatóságára vonatkozóan mindvégig támpontokat, adatokat szolgáltat.
- Kimutathatja már korán azt is, hogy a szoftver nem teljesíti a követelményeket.

Mindez a szoftver előre megtervezett ellenőrzésével, intenzív tesztelésével kell hogy járjon az egyes fázisokban. Jelen közlemény olyan módszerekkel foglakozik, amelyek a szoftver verifikálásával és validálásával kapcsolatosak, ahol a tesztelés mindkét tevékenység integráns része. Itt a hangsúlyt az úgynevezett *biztonságkritikus számítógéprendszerekre* helyezzük [3,7-9], ahol a legfontosabb kritérium a biztonságos működés, az élet veszélyeztetése nélkül, valamint nagyobb természeti vagy anyagi kár okozása nélkül. Az ilyen típusú számító rendszert azért vettük itt számításba, mivel ez igényli a legalaposabban tervezett és végrehajtott eljárásokat, más egyéb rendszerekkel összehasonlítva.

A cikk egy tesztmodellt mutat be, amely egy leképezési sémán alapul. A séma a bemeneti és kimeneti tartományok közötti egy-egy értelmű leképezést írja le, egy adott szoftverrendszerre vonatkozóan. Ebbe a tesztbemenetek és a hibaosztályok szintén beletartoznak. A tesztmodell mind a verifikációs, mind pedig a validációs sémákat is magában foglalja. A modell jelentősége abban áll, hogy megkönnyíti a tiszta különbségtételt a verifikációs és a validációs tesztek között, ami fontos és hasznos a teszttervezés és a tesztkiértékelés során. Végezetül a *formális módszerek* szoftvertervezési felhasználásának következményeivel és problémáival foglakozik a cikk.

2. Hibamodell és alapfogalmak

A szoftverhibák alapvető sajátsága, hogy a működés teljes időtartama alatt jelen vannak. A kérdés az, hogy a hatásuk miként nyilvánul meg a különböző szituációkban. A hibáknak két alaposztályát különböztetjük meg:

a) Specifikációs hibák: Azok a hibák, amelyek a fejlesztési ciklus kezdetén képződnek, és a szoftver téves működésében nyilvánulnak meg azáltal, hogy nem teljesülnek a valós felhasználói követelmények. A téves

működés tág értelemben tekintendő: Egyaránt következménye lehet a hibás, a hiányos, valamint a következetlen, ellentmondást tartalmazó specifikálásnak. Ezt a kategóriát nevezhetjük még *külső* vagy *felhasználói hibának* is.

b) Programozási hibák: Azoknak a hibáknak a széles köre tartozik ide, amelyeket a programozók követnek el, az előzőleg már specifikált szoftver tervezési és kódolási folyamatában. Ennek a kategóriának másik szinonímái: *belső* vagy *fejlesztési hibák*. Néhány lehetséges hibatípust ezekből az alábbiakban sorolunk fel:

– hibás funkcióteljesítés,
– hiányzó funkciók,
– adatkezelési hibák az adatbázis elérése során,
– kezdési és befejezési hibák,
– hibák a felhasználói interfészben,
– határértékek alá vagy fölé kerülés,
– kódolási hiba,
– algoritmikus hiba,
– inicializálási hiba,
– a vezérlési folyamat hibája,
– adatátviteli hiba,
– input-output hiba,
– programblokkok közötti versenyhelyzet,
– programterhelési hiba.

A kiválasztott hibamodell nagymértékben meghatározza azokat a ráfordításokat, amelyeket a tesztelési folyamatok megtervezésében és végrehajtásában kell kifejteni [10-13].

A legfontosabb mérlegelési szempontok:

- Lehetőségek a teszttervezés megvalósítására, a költségek figyelembevételével.
- Előzetes ismeretek azokról a hibajelenségekről, melyek az adott fejlesztési technikához kapcsolódnak.
- A rendelkezésre álló hardver- és szoftvereszközök szolgáltatási köre és teljesítménye.

A gyakorlati tapasztalatok alapján állítható, hogy a teljes fejlesztési költségek mintegy 50%-át teszik ki a tesztelési költségek. Ez magában foglalja tesztelési folyamatok megtervezését és végrehajtását is. A szoftver-technológia fejlődésével a rendszerek mérete és bonyolultsága egyre növekszik. Ennek következményeként állandó igény mutatkozik arra, hogy újabb és hatékonyabb tesztelési módszereket és eszközöket dolgozzunk ki.

A szoftverfejlesztésben fontos követelmény a teljes folyamat konzisztens módon való végigvitele. Az egyes fejlesztési állomások eredményei saját megjelenési formával rendelkeznek. A folyamat helyes végigvitele megköveteli, hogy az egyes reprezentációk közötti összhang bizonyítását. Végül is, azt kell bizonyítani, hogy a végső szoftver termék száz százalékig megfelel a kiindulási specifikációnak. Ennek a bizonyítási folyamatnak a megvalósítása oly módon történik, hogy az egymást közvetlenül követő fejlesztési fázisok közötti összhangot bizonyítjuk lépésenként. Ha egy fázisnál diszkrepancia mutatkozik, akkor azt addig kell módosítani, amíg az nem harmonizál az előző fázissal. A leírt tevékenységet, amelyben két egymást követő fázis közötti ekvivalenciát bizonyítjuk, verifikációnak nevezzük. Ennek pontosabb definíciója a következő:

Verifikáció: Az a folyamat, amelyben igazoljuk, hogy a szoftver egy fejlesztési fázisban teljesíti mindazokat a követelményeket, amelyeket az előző fázisban specifikáltunk.

A lépésenkénti verifikálás elvileg elegendő az ekvivalencia igazolására a kiindulási és a befejező fázis között. Mindazonáltal, mivel a teljes folyamat általában nem egzakt, vagyis nem biztosít abszolút bizonyítást egyik lépésben sem, egy külön önálló *végső verifikációra* is szükség van, amit a specifikáció és a végtermék között hajtunk végre.

Másfelől nézve, ennél a pontnál meg kell jegyeznünk, hogy a verifikációs folyamat tökéletes megvalósítása sem garantálná a végtermék tökéletes használhatóságát. Mindaz, amit garantálni lehet, a kiindulási specifikációval való ekvivalencia teljesülése. Abban az esetben, ha hiba vagy tökéletlenség volt a specifikációban, a végtermék nem fogja kielégíteni a felhasználói követelményeket. Emiatt szükség van még egy külön vizsgálati folyamatra is, amiben a terméket az eredeti rendeltetése szempontjából ellenőrizzük. Az ilyen jellegű bizonyítási folyamatot *validációnak* nevezzük. Ennek definíciója a következő:

Validáció: A szoftvernek olyan vizsgálata és kiértékelése, amiben meghatározzuk, hogy minden szempontból teljesíti-e a felhasználói követelményeket.

Egy biztonságkritikus rendszer esetben a következőket kell bizonyítani:

– funkcionálisan megfelelő működés,
– megfelelő teljesítmény,
– a biztonsági követelmények kielégítése.

A tökéletlen specifikálás következményeként leginkább a biztonság lesz veszélyeztetve. Mindezek után, a végső validációs eljárásban azt kell eldönteni, hogy a teljes rendszer biztonságos-e vagy sem. Ha valamilyen probléma adódik, akkor vissza kell térni a kezdeti specifikációhoz, és módosítani kell azt. A módosítás azzal jár, hogy újra kell tervezni a rendszert, a szükséges változtatások végigvitelével, minden egyes verifikációs fázist elvégezve, másrészt új validálásra is sort kell keríteni.

A verifikáció és validáció együtt kezelendő, teljes összhangban. Ezt a tényt a széles körben elterjedt *„V&V" eljárás* elnevezés is kifejezi [3]. A két fogalmat azonban néha összekeverik, annak ellenére, hogy lényegesen eltérő tevékenységeket jelölnek. A verifikáció annak ellenőrzése, hogy a program megfelel-e a specifikációjának. A validáció pedig annak eldöntésére irányul, hogy a megvalósított program teljesíti-e a felhasználó elvárásait.

A két tesztelési feladat egymástól lényegesen eltérő megközelítést igényel. A verifikáció esetében a teszteknek az előállított, rendelkezésre álló megjelenési formák közötti ekvivalenciát kell igazolniuk. Ehhez számos jól bevált teszttervezési módszert tudunk felhasználni [1,2,10-12]. A validációs tesztek előállításakor viszont arra kell törekedni, hogy egy biztonságkritikus rendszert többféle komplikált működési helyzetbe hozzunk, ellenőrzendő, hogy az mindegyik helyzetben ki-

állja-e a próbát, és nem okozhat károkat [3,7,8]. Az ilyen tesztek képzéséhez a felhasználási cél, valamint a biztonságosság elérése ad útmutatót.

3. Verifikációs és validációs modell

Amint már deklaráltuk, a szoftverhibák a programozás során alakulnak ki, illetve hibás vagy nem teljes specifikációból származnak. Az első kategória a nem megfelelő fejlesztési folyamat következménye, a második pedig a végső felhasználási követelményekkel való összhang hiánya. A következőkben ezt a két kategóriát rendre *fejlesztési hibának,* illetve *felhasználói hibának* fogjuk nevezni.

A szoftverhibák a program helytelen működésében nyilvánulnak meg, amikor a hibás kódszegmens végrehajtására kerül sor, annak a bemeneti adathalmaznak a hatására, amely előhozza az adott hibát. Ugyanakkor a kód többi része már jól működhet más bemenetekre nézve. Egy szoftverrendszert olyan egységnek tekinthetünk, ami *egy bemeneti halmazt egy kimeneti halmazra képez le.* Egy rendszernek nagyon sok bemeneti értéke lehetséges. Az egyszerűség kedvéért a bemeneti értékek kombinációját és szekvenciáját egyetlen bemeneti elemnek fogjuk tekinteni. A bemenetei értékekre adott válaszértékek képezik a kimeneti értékek halmazát.

Legyen a szoftver összes lehetséges bemeneti értékének halmaza **INPD** (input domain, – bemeneti tartomány), és az összes lehetséges kimeneti válasz halmaza **OUTD** (output domain, – kimeneti tartomány). Ezzel az INPD-nek a szoftver által történő leképezését az OUTD-re a következő formában fogjuk definiálni:

SWM (INPD) = OUTD. (1)

Az (1) reláció azt jelenti, hogy a rendszer működési tulajdonságai, vagyis az SWM leképezés (software mapping), határozzák meg az INPD és az OUTD elemei közötti megfelelést. A relációt ebben a formájában érvényesnek fogjuk tekinteni a szoftverfejlesztési ciklus bármelyik fázisában.

Jelöljük továbbá **INER**-rel (input errors) azon bemeneti értékek halmazát, amelyek hibás működést okoznak, míg a hibás válaszértékek halmaza legyen **OUTER** (output errors). Ezekre a halmazokra a következő leképezési reláció áll fenn:

SWM (INER) = OUTER. (2)

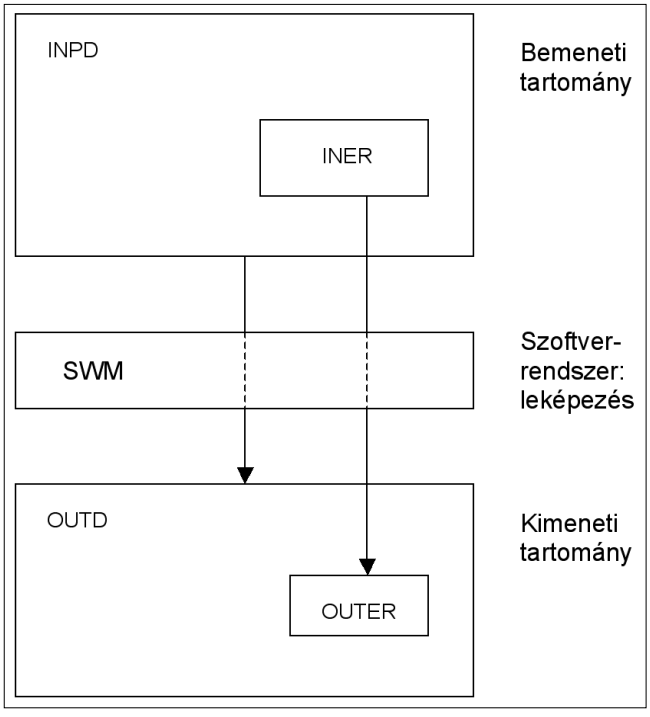
A fenti két leképezés sémáját az *1. ábra* mutatja be.

Ha a leképezési modellünkbe bevonjuk a hibafelfedő teszteket, akkor ez az INER és OUTER halmazok módosulását fogja eredményezni. Tegyük fel, hogy egy INERT elnevezésű teszthalmaz (input-error tests) alkalmas arra, hogy felfedje a még felderítetlen hibák egy részhalmazát. Ebben az esetben az INERT és INER szükségszerűen közös elemekkel kell hogy rendelkezzenek. Az elvárható következmény ekkor a felfedett hibák eltávolítása. Ez azt jelenti, hogy a javított szoftver egy módosított kimeneti tartományt fog produkálni, egy olyan OUTER részhalmazzal, amely már nem képviseli tovább a detektált és ily módon eltávolított hibákat. Ezek után a javított szoftverre vonatkozó új leképezési reláció

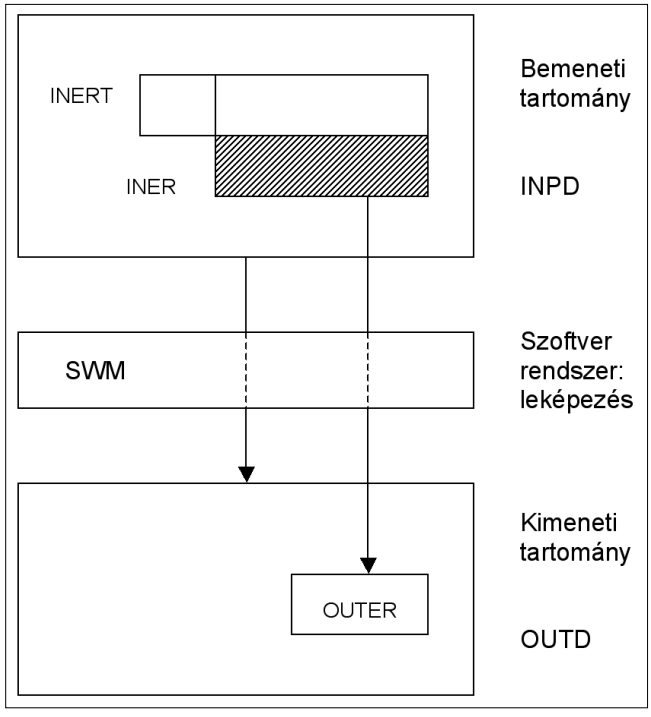
SWM (INER – INERT) = OUTER (3)

lesz, ahol a mínusz jel a halmazok közötti kivonást képviseli. Az ennek megfelelő leképezési séma a *2. ábrán* látható.

1. ábra Szoftver-leképezési séma hibák esetén



2. ábra Leképezési séma tesztelés után



Ennél a pontnál már rátérhetünk a verifikáció és validáció kérdésének vizsgálatára. Itt a következő jelöléseket vezetjük be:

- A fejlesztési hibákban megnyilvánuló bemenetek halmaza: **DEFI** (development-fault inputs). Ennek a halmaznak a kimeneti leképezése **DEFO** (development-fault outputs).
- A felhasználói hibákban megnyilvánuló bemenetek halmaza: **USFI** (user-fault inputs). Ennek a halmaznak a kimeneti leképezése **USFO** (user-fault outputs).
- A fejlesztési hibák detektálására készített tesztbemenetek halmaza, azaz a verifikációs tesztek halmaza: **VERT** (verification tests).
- A felhasználói hibák detektálására készített tesztbemenetek halmaza, azaz a validációs tesztek halmaza: **VALT** (validation tests).

Miután megvan az eljárásunk arra, hogy különböző input halmazokat output halmazokra képezzünk le, általános modellt tudunk adni a verifikáció és validáció leképezési sémájára, egy adott szoftverrendszerre vonatkozóan. A leképezési relációk a következők lesznek:

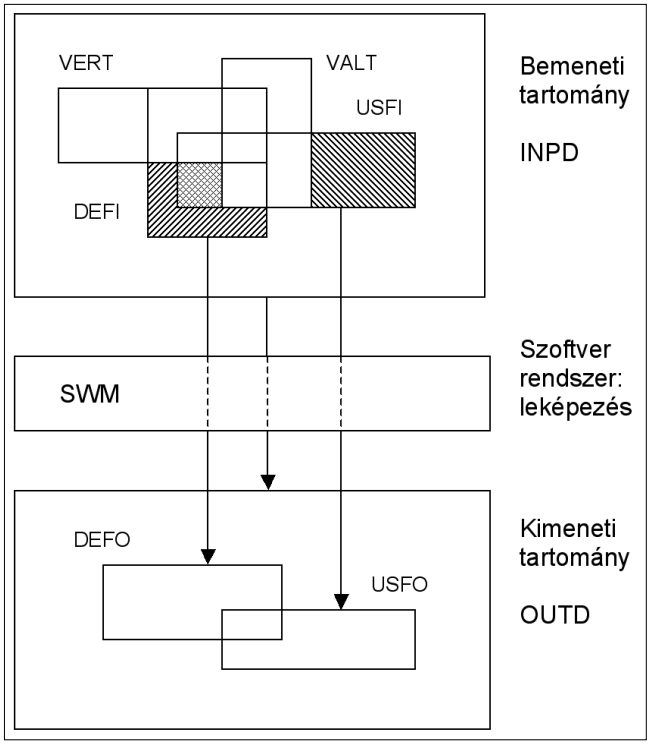
$$\mathbf{SWM\ (INPD) = OUTD.} \quad (4)$$

$$\mathbf{SWM\ (DEFI - (VERT \cup VALT)) = DEFO.} \quad (5)$$

$$\mathbf{SWM\ (USFI - (VERT \cup VALT)) = USFO.} \quad (6)$$

A fentiekben az (5) reláció a verifikációs leképezést, míg a (6) reláció a validációs leképezést fejezi ki. Ezekből a relációkból látható, hogy (5) a felfedetlen fejlesztési hibákat, a (6) pedig a felfedetlen felhasználói hibákat képviseli. A (4), (5) és (6) relációkat a *3. ábra* összetett leképezési sémája mutatja be.

3. ábra
Leképezési séma verifikációval és validációval



4. Formális módszerek használata

A formális módszerek matematikai technikát alkalmaznak a számítógépek hardverjének és szoftverjének specifikálásában, tervezésében és analízisében [7,14-15]. Ismeretes, hogy a biztonságkritikus rendszerek fejlesztésével kapcsolatos problémák egy jó része a specifikációs hiányosságokból ered [3]. A specifikációnak egyértelműnek, teljesnek, konzisztensnek és helyesnek kell lennie. Azok a dokumentumok, amelyek természetes nyelven íródtak, mindig ki vannak téve a félreértésnek. Ugyancsak nehéz elérni azt is, hogy ezek a dokumentumok a tervezendő rendszer teljes és helyes leírását képviseljék, vagy még azt is kimutatni, hogy ezek konzisztensek egymással.

A formális módszerek *formális nyelvek* használatán alapulnak, amelyeknek precíz és szigorú szabályaik vannak. Ez a sajátság lehetővé teszi, hogy a specifikálást olyan módon készítsük el, amely egyértelműen interpretálható. Mindemellett még az is lehetővé válik, hogy automatizáltan ellenőrizzük a specifikációt, azzal a céllal, hogy kihagyásokat és következetlenségeket találjunk benne, vagyis hogy bizonyítsuk a teljességet és a konzisztenciát.

Azok a nyelvek, amelyek ezt a célt szolgálják, a *rendszer-specifikációs nyelv,* vagy *formális specifikációs nyelv* elnevezést viselik. Használatuk számos potenciális előnyt kínál a fejlesztési ciklus mindegyik fázisában [7,14-21].

Az egyik legnagyobb előny az, hogy automatikus tesztek hajthatók végre az ilyen leírás alapján. Ez lehetővé teszi, hogy szoftver eszközökkel ellenőrizzünk bizonyos hibaosztályokat, másrészt hogy különböző leírásokat hasonlítsunk össze annak eldöntésére, hogy ekvivalensek-e. A terv különböző fázisai ugyanannak a rendszernek a leírásai, s így ezeknek funkcionálisan ekvivalensnek kell lenniük. Ha minden egyes reprezentáció megfelelő formában készült el, akkor közöttük egyenként bizonyítható lesz az ekvivalencia.

Mint látható, ez a folyamat nem más mint maga a verifikálás. A *4. ábra* azt mutatja, hogy minden egyes transzformációt annak ellenőrzése követ, hogy az helyesen hajtódott-e végre. Ez annak kimutatását jelenti, hogy egy adott fázis bemenetét adó leírás funkcionálisan ekvivalens azzal, ami a fázis kimenetén állt elő.

Ideális esetben a fent vázolt folyamat transzformációs eljárásai teljes mértékben automatizálva vannak, emberi beavatkozás nélkül, és mindegyik fázisban hibamentesen hajtódnak végre. Ha ez teljesül, akkor a kívülről származó verifikációs tesztsorozatok teljesen elhagyhatók lesznek. A tesztmodellünkben ez azzal jár, hogy

$$\mathbf{DEFI} = \varnothing, \qquad \mathbf{VERT} = \varnothing,$$

ahol a $\varnothing$ szimbólum az üres halmazt jelöli. Ha ezt a két eredményt az (5) relációba helyettesítjük, akkor

$$\mathbf{SWM}\ (\varnothing - (\varnothing \cup \mathbf{VALT})) = \mathbf{SWM}\ (\varnothing) = \mathbf{DEFO} = \varnothing \quad (7)$$

adódik.

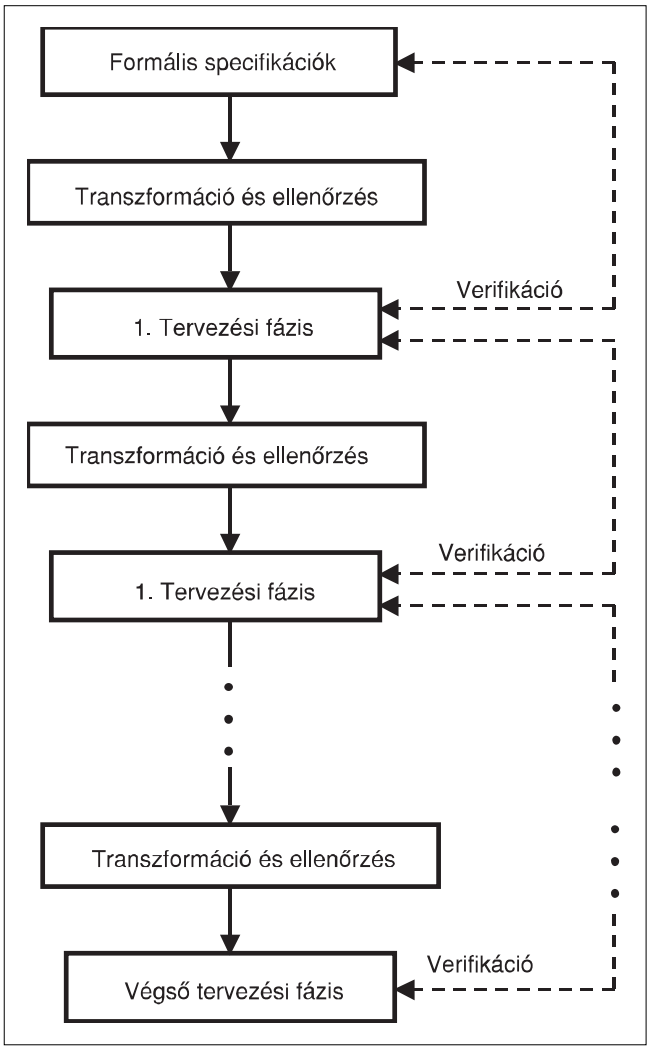
Másrészről, a szoftver kezdeti formális specifikálása mindig kézi úton történik, így emiatt a tervezési hibák sosem zárhatóak ki itt, még akkor sem, ha automatizált konzisztencia-ellenőrzés áll rendelkezésre. Ennek az oka az, hogy egy konzisztens terv még önmagában véve nem garantálja a felhasználói követelmények maradéktalan teljesítését.

Hasonlóképpen, nincsen garancia a biztonsági követelmények teljes mértékű kielégítésére sem. Következésképpen a külső validációs tesztelés alkalmazására mindig szükség van. A fenti ideális eset figyelembevételével a (6) reláció redukált alakja a következő lesz (8):

SWM (USFI – (∅ ∪ VALT)) = SWM (USFI – VALT) = USFO.

Végezetül hangsúlyoznunk kell, hogy a vázolt eljárás nem tekinthető csodaszernek. Jóllehet a formális módszerek használata számos előnnyel jár, jelentős számú megszorítást is hordoz magában a fejlesztési módszerekre nézve. Mivel egy lánc erősségét a leggyengébb eleme határozza meg, a maximális haszon elérése érdekében szükségessé válik a formális bizonyítások elvégzése a fejlesztés minden egyes fázisában. Ez komoly kihatásokkal van a projekten belül alkalmazható tervezési módszerekre. A legtöbb esetben a teljesen automatizált végrehajtás nem alkalmazható, így továbbra is szükség van a jól képzett tervezők szoros együttműködésére és beavatkozására. Jelenleg a formális módszerek legfőbb előnye abban van, hogy a tervezési és verifikálási folyamatok nagyobb megbízhatóságú végrehajtását teszik lehetővé. A következőkben néhány konkrét problémát sorolunk fel a formális módszerekkel kapcsolatosan:

4. ábra Formális módszerek alkalmazásának folyamata



1) A gyakorlati alkalmazások területén eddig még kevés tapasztalat gyűlt össze. A fő gondot a számítási komplexitás jelenti, amit igen nehéz előzetesen meghatározni. A komplexitásra vonatkozóan az tapasztalható, hogy egyes algoritmusok az NP-teljes feladatok osztályába tartoznak. Mint ismeretes, az NP-teljes feladatok számítási komplexitása olyan, amire várhatóan nem létezik felülről korlátozó véges fokszámú polinom, ahol a polinom változója a feladat méretét képviseli. Ez valójában azt jelenti, hogy a számítási lépések száma véges, de megjósolhatatlanul nagy.

2) A biztonságkritikus rendszerek biztonságigazolási folyamata, vagyis a validációs folyamat elvileg nem automatizálható teljes mértékben. Ez azért van így, mert ebben az esetben a tesztelésnek mindig kell hogy legyenek olyan elemei, amelyek kívül esnek a rendszerspecifikáción, vagyis ezek a kiegészítő elemek függetlenek a specifikációtól. Az alapvető ok az, hogy azok a biztonsági problémák, amelyek a hiányos vagy téves specifikációból erednek, egyedül csak ezen a módon fedezhetők fel. Természetesen ez az állítás nem csak a klasszikus módszerekre vonatkozik, hanem a formálisokra is.

3) Jelenleg nem áll rendelkezésre olyan egzakt formális specifikálási módszer, amellyel magának a biztonságnak az elvét tudnánk számításba venni. Más szóval, a biztonsági célokat közvetlenül szolgáló formális specifikációs módszerek még nincsenek kidolgozva.

A jelenlegi helyzet és az egyre fokozódó igény további jelentős kutatási és fejlesztési erőfeszítéseket követel, annak érdekében, hogy a formális módszerek hatékonyabbá váljanak a gyakorlatban. Hogy jelezzük az irányt, befejezésül néhány nyitott problémát, illetve kérdést vetünk fel az alábbiakban:

- Lehetséges-e formalizálni a biztonság szintet, amit el kell érni a tervezési folyamatban?
- Ha igen, ez milyen módon valósítható meg?
- Lehetséges-e meghatározni egy hibatűrő rendszer biztonságát kvantitatív módon?
- Lehetséges-e közvetlen kapcsolatot megteremteni egy hibatűrő struktúra és a formális tervezés között?

5. Befejező megállapítások

Ez a közlemény a szoftvertesztelésre vonatkozóan egy általános modellre ad javaslatot. A modell egy leképezési sémán alapszik, amely a bemeneti és kimeneti tartományokat foglalja magában, valamint a különböző tesztelési feltételeket és hibalehetőségeket.

A bemutatott elv fő célja az, hogy világosan megkülönböztesse a verifikációra és a validációra szolgáló teszteket, ami különösen fontos és hasznos a biztonságkritikus rendszerek esetében. A két tesztelési feladat ugyanis egymástól lényegesen eltérő megközelítést igényel. Mint láthattuk, az itt bemutatott modell alkalmazható a formális módszereken alapuló szoftverfejlesztésnél is. Ugyanakkor a cikk kimutatta azt is, hogy a validációs eljárások velejáró elméleti korlátozásokkal bírnak, amikor formális specifikációt alkalmazunk.

A biztonság-orientált informatikai rendszerek tervezésében ma már döntő szerepet játszik a szoftver eszközök felhasználása. A szoftver meghatározó szerepe a rendszerek üzemeltetése során is érvényesül, a bennük megvalósított komponenseken keresztül. Mindezek miatt egyre inkább növekszik a megbízható szoftver előállítására fordítandó kutatási-fejlesztési tevékenységek fontossága.

Végezetül, mint ismeretes, az informatikai rendszerek hardver összetevője szintén szigorú és pontos fejlesztési és gyártási technológiát igényel. Ez az irányelv mindenek előtt a biztonságkritikus rendszerekre érvényes, ahol a hardver és szoftver együttes tervezése széles körben használatos megközelítés [3,5].

Ebben a megközelítésben a végleges megosztás a két szféra funkciói között a tervezési folyamat során dől el. Egy hardver rendszer biztonságos és megbízható működése szintén megköveteli a verifikálást és a validálást, mind a tervezési, mind pedig a gyártási ciklusban [16].

Ami a fentiekben bemutatott tesztmodellt illeti, belátható, hogy az, bizonyos pótlólagos megfontolásokkal, kiterjeszthető a hardver rendszerek kezelésére is. Egy ilyen jellegű kiterjesztés hibamodellje a hardver tervezési hibáit, gyártási hibáit, valamint üzemeltetési hibáit foglalja magában.

Irodalom

[1] Ian Sommerville:
Software Engineering, 6th Ed.,
Addison-Wesley Publ. Company, Inc., USA, 2001.

[2] Roger S. Pressman:
Software Engineering, 5th Ed.,
McGraw-Hill Book Company, USA, 2001.

[3] Neil Storey: Safety-Critical Computer Systems,
Addison-Wesley-Longman, Inc., New York, 1996.

[4] Marvin V. Zelkowitz:
Role of Verification in the Software Specification Process,
Advances in Computers,
(Editor: Marshall C. Yovits), Vol. 36, pp.43–109.,
Academic Press, Inc., San Diego, 1993.

[5] Jean-Michel Bergé, Oz Levia, Jacques Rouillard:
Hardware/Software Co-Design and Co-Verification,
Kluwer Academic Publishers, Dordrecht, NL, 1997.

[6] Nicolas Halbwachs, Doron Peled (Editors):
Computer Aided Verification,
Lecture Notes in Computer Science, Vol. 1633,
Springer-Verlag, Berlin, 1999.

[7] József Sziray, Balázs Benyó, István Majzik,
András Pataricza, Júlia Góth, Levente Kalotai,
Tamás Heckenast: Quality Assurance and
Verification of Software Systems, (In Hungarian),
Széchenyi College, Budapest Technical and
Economical University, University of Veszprém, 2000.

[8] Nancy G. Leveson:
Safeware: System Safety and Computers,
Addison-Wesley Publ. Company Inc., USA, 1995.

[9] Dhiraj K. Pradhan:
Fault-Tolerant Computer System Design,
Prentice-Hall, Inc., USA, 1996.

[10] Glenford J. Myers:
The Art of Software Testing,
John Wiley & Sons, Inc., New York, 1979.

[11] Cem Kaner, Jack Falk, Hung Quoc Nguyen:
Testing Computer Software,
Van Nostrand Reinhold, Inc., New York, 1993.

[12] Boris Beizer:
Black-Box Testing, Techniques for
Functional Testing of Software and Systems,
John Wiley & Sons, Inc., New York, 1995.

[13] Péter Várady, Balázs Benyó:
A Systematic Method for the Behavioural Test and
Analysis of Embedded Systems, IEEE International
Conf. on Intelligent Engineering Systems, Proc.,
pp.177–180., Portoroz, Slovenia, Sept. 17-19, 2000.

[14] Michael J. C. Gordon:
Programming Language Theory and its Implementation,
Prentice Hall International Ltd, Great Britain, 1988.

[15] Constance Heitmeyer, Dino Mandrioli (Editors):
Formal Methods for Real-Time Computing,
John Wiley & Sons, Inc., New York, 1996.

[16] Balázs Benyó, József Sziray:
The Use of VHDL Models for Design Verification,
IEEE European Test Workshop, Proc.,
pp. 289–290., Cascais, Portugal, May 23-26, 2000.

[17] Martin Fowler, Kendall Scott:
UML Distilled: Applying the Standard Object
Modeling Language,
Addison-Wesley-Longman, Inc., USA, 1997.

[18] Mark Priestley:
Practical Object-Oriented Design with UML,
McGraw-Hill Publishing Company, GB, 2000.

[19] Csaba Szász, József Sziray:
Run-Time Verification of UML State-Chart
Implementations, IEEE Intern. Conf. on Intelligent
Engineering Syst., Proc., pp.355–358.,
Portoroz, Slovenia, Sept. 17-19, 2000.

[20] Zsigmond Pap, István Majzik, András Pataricza,
András Szegi:
Completeness Analysis of UML Statechart Specific.,
IEEE Design and Diagnostics of Electronic Circuits
and Systems Workshop, Proc., pp.83–90,
Győr, Hungary, April 18-20, 2001.

[21] Eric J. Braude: Software Engineering,
An Object-Oriented Perspective,
John Wiley & Sons, Inc., New York, 2001.

Objektumorientált környezetben készült biztonságkritikus szoftverrendszerek verifikálása

BENYÓ BALÁZS

Széchenyi István Egyetem, Informatika Tanszék
benyo@sze.hu

Kulcsszavak: biztonságkritikus rendszer, szoftver verifikáció, iteratív tesztgenerálás, regressziós tesztelés

Annak érdekében, hogy modern, objektumorientált (OO) technológia előnyeit kihasználó szoftverfejlesztés lehetővé váljon OO környezetben alkalmazható szoftver verifikációs módszerekre és a módszerek egyszerű megvalósítását lehetővé tevő fejlesztői környezetre van szükség. A cikkben egy olyan szoftver verifikációs eljárásokat és az eljárásokat támogató, ill. megvalósító keretrendszer kerül bemutatásra, mely lehetővé teszi objektumorientált rendszerek alapos tesztelését és támogatja az auditáláshoz szükséges dokumentáció előállítását. A keretrendszer vasútirányító szoftverek auditálásához készült.

A biztonságkritikus rendszerek fejlesztésekor különösen nagy gondot kell fordítani a rendszerek szolgáltatásbiztonságának igazolására. Különböző szakterületeken (vasút, egészségügy, légi közlekedés stb.) szakterületi szabványok rögzítik annak szabályait, hogy milyen vizsgálatokat kell az adott területen alkalmazott rendszer használatbavétele előtt elvégezni. Ezek a szabványok definiálják azt a kritériumrendszert, mely alapján az adott területen használt rendszert a szolgáltatásbiztonság szempontjából minősíteni vagy auditálni lehet.

Az alkalmazott rendszerek méretének és bonyolultságának növekedésével ezek a szabványok egyre kevésbé alkalmasak a megfelelően részletes szabályozásra, így egyre kevésbé alkalmazhatóak közvetlenül a gyakorlatban. Míg az elsősorban mechanikus és elektronikus hardver elemeket tartalmazó rendszerek esetén viszonylag könnyen lehetett általános szabályokat definiálni a rendszerek szolgáltatásbiztonságának kimerítő vizsgálatára, úgy az elsősorban szoftver komponensekből álló rendszerek esetén ezek a szabványok megmaradnak az általánosságoknál.

A hiányos, gyakran túl általános szabályozás a fejlesztők felelősségévé tette a szoftverrendszerek megfelelő módszerekkel, megfelelő mértékben történő verifikálását és validálását. A gyakorlatban ez azt eredményezte, hogy a biztonságkritikus szoftverek előállítói – óvatos megközelítéssel – csak kiforrott, régóta használatos szoftverfejlesztési technológiát és környezetet alkalmaztak, melyek esetén rendelkeztek a megfelelő módszertani és technológiai háttérrel a rendszerek teszteléséhez. Így gyakran előfordul, hogy biztonságkritikus szoftverrendszereket még ma is strukturált megközelítésben, procedurális programnyelveken fejlesztenek.

1. Célkitűzések

Az OO rendszerek verifikálása és tesztelése során a legnagyobb problémát a rendszerek korlátozott megfigyelhetősége okozza [3,6,8]. Az OO rendszerek korlátozott megfigyelhetősége az OO nyelvek természetéből adódik. Az egységbezárás (encapsulation), illetve az adatrejtés (data hiding) a OO programozási paradigma alapelvei, melyek elősegítik a kód-újrafelhasználást és a hatékony programfejlesztést. Ezt a problémát ügyes, a tesztelés szempontjait is figyelemben vevő tervezéssel, illetve implementációval legfeljebb csökkenteni lehet, de elkerülni nem.

A cikkben bemutatott kutatás-fejlesztési munka célja olyan verifikációs módszerek kidolgozása volt, melyek lehetővé teszik nagyméretű és bonyolult OO szoftverrendszerek verifikálását és ugyanakkor alkalmazhatóak a gyakorlatban. Valós méretű problémák esetén történő alkalmazhatósága a módszereknek kitüntetett fontosságú volt, mert az irodalomban publikált módszerek jelentős része a gyakorlatban nem állta meg a helyét [1]. A cikkben ismertetett módszereket verifikációs keretrendszer formájában implementáltuk annak érdekében, hogy gyakorlatban is kipróbáljuk őket. A gyakorlatban történő alkalmazhatóságon felül a módszerek kidolgozásakor, illetve a keretrendszer fejlesztésekor a következő célokat tűztük ki:

- Tegye lehetővé nagyméretű és összetett OO rendszerek tesztelését.
- A rendszer verifikálásának támogatása a fejlesztés minden fázisában.
- A rendszer végső auditálásának támogatása.
- A rendszerfejlesztők és tesztelők munkájának felhasználóbarát támogatása.
- Különböző tesztelési módszerek támogatása függetlenül a tesztelt rendszer jellemzőitől.

2. Módszerek

Minden klasszikus tesztelési stratégiának (hierarchikus tesztelés: top-down, bottom-up tesztelés; izolációs tesztelés stb.) és tesztelési módszernek (határérték tesztelés, ekvivalencia partícionálás, úttesztelés stb.) van olyan előnyös tulajdonsága, mely az adott tesztelési

stratégiát vagy módszert optimálissá teszi egy-egy speciális felépítésű, vagy adott tulajdonságokkal rendelkező rendszer tesztelésekor [2,7,8]. A gyakorlatban sajnos a tesztelt alkalmazások nem homogén felépítésűek ebből a szempontból, vagyis az egyes módszerek csak a tesztelt rendszer egy jól meghatározható részére lesznek hatékonyak.

Felismerve ezt az inhomogén tulajdonságát a valós rendszereknek a tesztelési környezet magját úgy alakítottuk ki, hogy az egyes, környezet által támogatott módszerek opcionálisan választhatóak, azokat nem kötelező alkalmazni minden rendszerkomponens esetén. A tesztelési környezet ebből a szempontból nyitott, könnyen kiegészíthető bármilyen további módszert támogató komponenssel, valamint a különböző tesztelési módszerek kombinálhatóak egy adott részrendszer tesztelésekor.

A következőkben röviden ismertetjük azokat a tesztelő, illetve tesztelést támogató módszereket, melyek implementálásra kerültek a tesztelő keretrendszerben.

A. Objektumok becsomagolása (object wrapping)

OO rendszerek verifikálásakor kritikus kérdés, hogy hogyan lehet megoldani a rendszer viselkedésének megfigyelését. A problémát az okozza, hogy OO alkalmazásokban az osztályok és objektumok tagváltozói, illetve tagfüggvényeinek elérhetősége korlátozott, azokat csak a kód jól meghatározott részeiből lehet olvasni vagy meghívni. Az OO rendszereknek ezt a lehetőségét, illetve tulajdonságát a jól tervezett OO alkalmazások igen intenzíven használják, mert ez segíti a kódújrafelhasználást [1,13].

Annak érdekében, hogy ellensúlyozzuk az OO rendszerek ezen tesztelést megnehezítő tulajdonságát szükséges, hogy a tesztelő keretrendszer támogassa a tesztelt rendszerben lévő objektumok viselkedésének megfigyelését. Ennek egyik lehetséges módja olyan megfigyelő osztályok készítése, amelyek olyan viszonyban (például C++ esetén friend) vannak a megfigyelt osztállyal, mely megengedi a külvilág számára nem elérhető adatok elérését. Ez az út sajnos az esetek többségében nem járható, mert csak a megfigyelt rendszer megváltoztatásával valósítható meg.

Az általunk készített keretrendszerben az objektumok viselkedésének megfigyelését a klasszikus rendszerekben alkalmazott becsomagolás (wrapping) módszer OO rendszerek osztályainak megfigyelésére kidolgozott változatával támogattuk. A módszer működését az *1. ábra* szemlélteti.

A tesztelés megkezdésekor minden rendszerbeli osztályhoz egy úgynevezett csomagoló (wrapper) osztályt hozunk létre. Ezek a csomagoló osztályok a teszteléshez kapcsolódó feladatok támogatására szolgálnak. A tesztelő rendszer ezen csomagoló osztályokon keresztül fogja a tesztelt rendszer objektumait elérni. A csomagoló osztály rendelkezik minden olyan külvilág által elérhető függvénnyel, mint a megfigyelt rendszerben levő párja. Ezen függvények törzse két funkciót valósít meg:

– a függvényhívás tényének és paramétereinek elmentése a tesztelés eredményét rögzítő adat fájlba;
– az eredeti megfigyelt rendszerben levő osztály megfelelő függvényének meghívása.

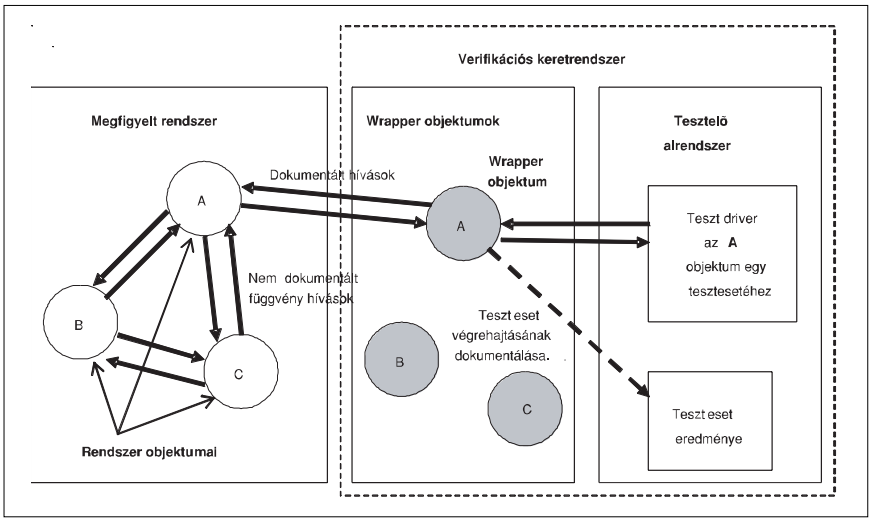
A rendszer működése ezek után egyszerű. A teszteset végrehajtásakor a tesztelő rendszer ezen csomagoló osztályokat fogja létrehozni és használni. A csomagoló osztályok fogják automatikusan instanciálni a megfigyelt rendszerben levő párjukat. Amikor a tesztelő környezet teszteli az egyes tagfüggvényeket, akkor a csomagoló osztály megfelelő függvényét fogja meghívni, ami automatikusan dokumentálja a hívást, és a paramétereket, majd aktivizálja az eredeti függvényt. A csomagoló függvény képes a tesztelt függvény visszatérési értékét is a teszteredmény fájlba elmenteni.

Természetesen ez a módszer nem fogja tudni a rendszeren belül történő hívásokat dokumentálni, azonban szisztematikus módszert ad az objektumok tesztelés során történő megfigyelésére. Az, hogy a rendszeren belül történő hívásokat nem tudjuk megfigyelni a hibadiagnosztikát megnehezíti. Ez a probléma azonban részben kiküszöbölhető az osztályok megfelelő sorrendben történő tesztelésével. Ha figyelembe vesszük az objektumok használati sorrendjét a rendszer természetes működése során, lehetséges olyan tesztelési stratégia kialakítása, mely esetén a egy adott teszteset végrehajtásakor az aktuálisan tesztelt hívás csak már tesztelt rendszeren belüli függvényhívásokat használ.

B. Tesztelés alaposságának mérése

Minden tesztelés során szükséges a tesztelés folyamatát megtervezni és irányítani, definiálni, hogy mely teszteseteket kívánjuk a tesztelés egy adott fázisában végrehajtani a tesztelt rendszeren. A tesztelés során ehhez általában valamilyen kvantitatív mérőszámot hasz-

1. ábra Csomagoló (wrapper) osztályok használata



nálunk, mely leggyakrabban egy arányszám, amely azt hivatott kifejezni, hogy a tesztelt rendszerünk mekkora részét teszteltük le tesztelés adott fázisában [7]. Számos ilyen arányszám létezik, azonban a gyakorlatban az utasítás lefedettség (statement coverage) mérőszám használható a legáltalánosabban. Az utasítás lefedettség használatának előnyei:

- Egyszerű az utasítás lefedettség mérése, és egyszerű az eredmény interpretálása.
- Összehasonlítva más mérőszámokkal, nem kötődik szorosan egyetlen tesztelési megközelítéshez, vagy módszerhez sem [4].

A tesztelési keretrendszerben az utasítás lefedettséget használtuk a tesztelés alaposságának jellemzésére.

C. Iteratív tesztelés

A modern OO rendszerfejlesztési metodikák alapján történő szoftverfejlesztés esetén az alkalmazásokat iteratív fejlesztési fázisok eredményeként állítjuk elő [13]. A rendszer által megvalósítandó funkciókat csomagokra osztjuk. A különböző csomagokban definiált funkciókat egymás után implementáljuk. Ennek megfelelően az egyes iterációk eredményeként előállított szoftver verziók az előző fázisokban előállított verziók kiterjesztett változatai lesznek [10].

A tesztelésnek illeszkedni kell ehhez az iteratív fejlesztési metodikához. Az iteratív fejlesztésnek két fontos következménye van a tesztelés szempontjából:

- Erősen támogatni kell a regressziós tesztelést, vagyis a tesztesetek ismételt végrehajtását, egy adott komponens újratesztelését.
- Támogatni kell a tesztesetek halmazának egyszerű bővítését.

A keretrendszer a tesztesetek halmazának egyszerű bővíthetőségét a következő alfejezetben ismertetésre kerülő hierarchikus teszteset azonosítókkal támogatja. A tesztesetek egyszerű megismételését a teszt driverek kódjának a forrásprogramként történő definiálásával oldottuk meg, mely definíció célszerűen a tesztelt rendszer fejlesztési környezetéhez illeszkedik. Ennek részleteit a következő alfejezetek tartalmazzák.

Az iteratív fejlesztés közvetlen támogatásán felül az iteratív fejlesztés gondolatát közvetlenül is alkalmaztuk a tesztesetek definiálásakor. Mivel a tesztelő környezet már fel volt készítve lépésenként növekvő teszthalmazok kezelésére és ismétlődő végrehajtására, kidolgoztuk az iteratív tesztelési módszerét, mely illeszkedik a klasszikus tesztelés gyakorlatához.

Az iteratív tesztelés alapötlete igen egyszerű. Egy adott funkcionalitású komponens tesztelését több fázisra bontjuk. Az egymást követő fázisokban a tesztelésnek egyre szigorúbb feltételeknek kell eleget tennie. Ennek megfelelően az egyes fázisokban használt tesztesetek halmazai – hasonlóan a szoftver verziókhoz – egyre bővülő halmazok, egymás kiterjesztett változatai lesznek.

Ennek a többfázisú tesztelésnek a potenciális előnye lehet a fejlesztés felgyorsítása. Elvileg egy adott fejlesztési fázis csak a korábbi fejlesztési fázis tesztelése után kezdődhet meg. A tesztelés azonban igen időigényes feladat, főleg biztonságkritikus rendszerek esetén, amikor a tesztelésnek igen szigorú követelményeknek kell eleget tennie. Iteratív tesztelés esetén elképzelhető, hogy a tesztelés valamelyik korai fázisának lezárása után elkezdődhet a termék következő fejlesztési fázisa. Ugyan a rendszerben potenciálisan maradnak még felderítetlen hibák, de azok száma viszonylag kicsi, így javításuk nem lesz olyan költséges a már részben továbbfejlesztett kódban, hogy ne érje meg ezt az árat megfizetni a fejlesztés lényeges felgyorsításáért.

A tesztelési keretrendszerben a tesztelésnek három iteratív fázisát definiáltuk:

– előzetes tesztelés,
– modul tesztelés,
– integrációs tesztelés.

Az egyes iterációk elnevezése nem véletlenül hasonló a klasszikus tesztelésben használt tesztelési fázisok elnevezéséhez. Az egyes tesztelési iterációk hasonló szerepet töltenek be az egyes fejlesztési fázisokban előállított szoftverek tesztelésekor, mint a lineáris, nem iteratív rendszerfejlesztés során alkalmazott tesztelési fázisok (modul tesztelés, integrációs tesztelés).

Definíciónk szerint tesztelő keretrendszerben a három iteratív tesztfázisban előállított teszthalmaznak a következő kritériumokat kell teljesíteni:

Előzetes tesztelés:

A szoftver legfontosabb, leggyakrabban használt funkcióit kell letesztelni. Nem szigorú kritérium a száz százalékos utasítás-lefedettség. A gyakorlatban 60-95%-os lefedettséget értek el az ebben a fázisban kidolgozott teszthalmazhoz tartozó tesztesetek.

Modul tesztelés:

A cél a tesztelt szoftver adott fejlesztési fázisban definiált komponenseinek önállóan történő alapos tesztelése. Követelmény a 100%-os utasítás lefedettség.

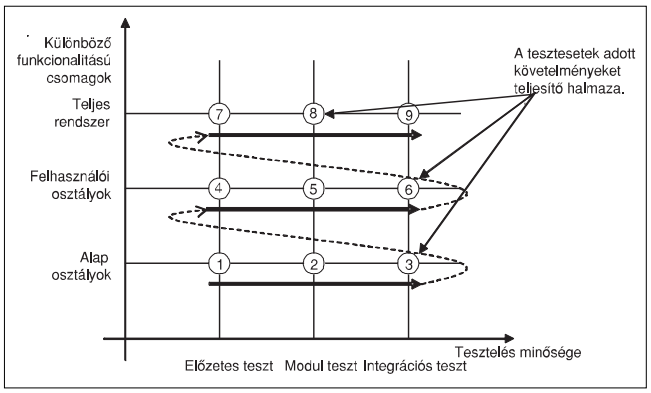
Integrációs tesztelés:

A cél a tesztelt szoftver adott fejlesztési fázisban definiált komponensei közötti kommunikáció alapos tesztelése. A komponenseket ebben a tesztelési fázisban egymással kommunikáló egységeknek tekintjük, és a tesztelés célja a köztük levő interfészek alapos tesztelése. Követelmény a 100%-os utasítás lefedettség.

A *2. ábra (a következő oldalon)* iteratív tesztelés módszerének alkalmazása esetén előállított tesztesethalmazokat mutat. Az ábrán a példaként vett szoftvernek három iteratív fejlesztési fázisban előállított verzióját láthatjuk: *Alap osztályok, Felhasználói osztályok, Teljes rendszer.* Minden egyes szoftver verziónál mind a három, fent definiált iteratív tesztelési fázisban előállított teszthalmazt szemléltettük. A teszthalmazokra írt számok, illetve a nyilak a teszthalmazok előállításának sorrendjét mutatják.

D. Tesztesetek hierarchikus számozása

A tesztesetek az adott rendszernek egy-egy jól definiált tulajdonságát verifikálják. A teszteseteket egyértelműen azonosítani kell a tesztelés során, az azonosí-



2. ábra
Különböző teszteset halmazok iteratív tesztelés esetén

tásra a teszteset azonosító (ID) szolgál. A teszteseteket elláthatjuk manuálisan ezzel az azonosítóval, azonban egy több ezer teszteset esetén már igen nehézkes. Egyszerűbb módja a teszteset azonosításnak, ha a teszt driver (teszt vezérlő) kódrészletek önmaguk generálják ezt az azonosítót a teszthalmazban levő elhelyezkedésük alapján.

Mivel a tesztesetek megvalósítását, a teszt driver (teszt vezérlő) kódrészleteket szekvenciálisan tároljuk, ahol célszerűen egymás után helyezkednek el egy adott osztály vagy komponens adott funkcióját verifikáló teszt driverek, nehézséget okozhat új tesztesetek beszúrása a teszthalmazba anélkül, hogy megváltozzon a beszúrt teszteset utáni tesztesetek azonosítója.

Ennek a problémának a megoldására vezettük be a hierarchikus teszteset azonosítók használatát. A teszteset azonosítók pontokkal aláosztott számsorozatok, pl. 1.15.24. Ebben az esetben, ha a teszthalmazba új tesztesetet kell beszúrni nem kell másra ügyelni, minthogy az újonnan beszúrt teszteset új aláosztást kezdjen. Tehát ha a 1.15.24-es teszteset után már volt 1.15.25-ös teszteset definiálva, akkor a kettő teszteset közé 1.15.24.1, 1.15.24.2 stb. számú teszteseteket szúrjuk be. Az azonosítók generálása automatikusan történt a tesztkörnyezet segítségével. Ha a teszteset definiálója új aláosztást akart kezdeni egy egyszerű deklarációt szúrt a teszteset teszt driverének első sora elé, és a rendszer automatikusan új azonosító hierarchiát kezdett.

E. Tesztesetek definiálása

A tesztesetek végrehajtását végző teszt driverek a tesztelt rendszer környezetéhez igazodva készültek. A teszt driverek forráskódját lefordítva egy olyan végrehajtható alkalmazást generáltunk, mely képes volt a teszteseteket végrehajtani. Mivel a teszt driverek tartalmazták a tesztesetek definícióját, generálták az azonosítóit, így – megfelelő paraméterezéssel – ez a program képes volt a teszteset leírásokat, specifikációkat is generálni. Természetesen paraméterezhető volt, hogy a tesztelő alkalmazás mennyire részletes tesztelési eredmény kimenetet készítsen: minden függvényhívás paraméterei szerepeljenek, vagy csak a tesztesetek eredménye. A teszt driverek tartalmazták a teszteset elfogadásának kritériumait, tehát maguk ellenőrizték a teszteset végrehajtásának hibás, illetve hibátlan voltát.

A tesztelő keretrendszer megvalósítása során egy C++ környezetben fejlesztett alkalmazást teszteltünk, így a keretrendszer is C++ környezetben készül el. A keretrendszer magja tartalmazza az összes olyan osztály, adatszerkezet definícióját, mely szükséges a tesztesetek egyszerű definiálásához.

Egy tipikus teszteset definíció három részből áll:

- Teszteset leírásából, specifikációjából.
  - TS_REQ: A tesztelt követelmény, illetve funkció leírása, mely a rendszer követelmény specifikációjában szerepelt.
  - TS_DESC: Annak a módszerének a leírása, amivel a követelményt teszteli a teszteset.
  - TS_EXPOUT: A kívánt, helyes működés során várt eredmény leírása.
- A tesztesetet végrehajtó kódrészletből, a teszt driverből.
- A teszteset végrehajtása után, az eredmény vizsgálatához szükséges ellenőrzések definíciójából. (TS_ASSERT).

A *3. ábrán* egy tipikus teszteset definíciót láthatunk. A TS_CASE_BEGIN(2) a tesztesethez tartozó teszt driver definíciójának elejét a TS_CASE_END a végét jelöli. A TS_CASE_BEGIN után a (2) azt jelöli, hogy a teszteset azonosítójában 2 aláosztás kell hogy szerepeljen.

```
TS_CASE_BEGIN(2);
TS_REQ("getNumTimersUsed: Returns the number of actual registered framework timers in the
        process.");
TS_DESC("Calling getNumTimersUsed(), no timers.\n"
          "Function returns the number of existing timers, i.e. 0.");
TS_EXPOUT( "Returns the number of actual registered framework timers in the process.");
TS_METHOD(TS_BOUND);
TS_CLASS_POS;
TS_TRY
{
        TS_ST_TimerControl          test_timerControl(timerControl);
        UInt32 testNumTimersUsed = test_timerControl.getNumTimersUsed();
        // check the number of existing timers
        TS_ASSERT( testNumTimersUsed == g_initedTimers );
}
TS_CATCH_ASSERT_FALSE;
TS_CASE_END;
```

3. ábra
Egy egyszerű teszteset definíciója

A 3. ábrán definiált teszteset végrehajtásakor generált teszt eredmény fájl adott tesztesethez tartozó részét a *4. ábrán* láthatjuk. A teszteset azonosítója 1.2.2, ami két aláosztást tartalmaz a 3. ábrán látható deklarációnak megfelelően. Az teszteset eredményének első részében láthatjuk a teszteset leírását a 3. ábrán definiált sorrendben. Ezután a teszteset végrehajtásának eredménye látható, ami ebben az esetben pozitív, tehát a tényleges eredmény azonos a várt eredménnyel.

```
TEST_CASE:              #1.2.2  (106)
TEST_REQUIREMENT:       getNumTimersUsed: Returns the number of actual
                        registered framework timers in the process.
TEST_DESCRIPTION:       Calling getNumTimersUsed(), no timers.
                        Function returns the number of existing timers, i.e. 0.
TEST_EXPECTED_OUTPUT:   Returns the number of actual registered framework
                        timers in the process.
TEST_METHOD:            Test case execution from boundary value analysis
TEST_CLASSIFICATION:    POSITIVE
-------------------
TEST_FUNCTION_CALL:     ST_TimerControl::getNumTimersUsed()
TEST_ASSERTED_TERM:     testNumTimersUsed == g_initedTimers
TEST_ASSERT_RESULT:     OK
TEST_RESULT:            PASSED
```

4. ábra
A 3. ábrán bemutatott teszteset végrehajtásakor keletkező kimenet

A tesztelő keretrendszer tartalmazza a az utasítás lefedettség mérésére és a tesztelési eredmény fájl elemzésére, statisztika készítésére szolgáló komponenseket is. Egy ilyen kiértékelés eredményét láthatunk az *5. ábrán*, mely az utasítás lefedettség mértékét mutatja. Az utasítás lefedettség a kódsorok végrehajtásának megfigyelésén alapult, mely információt a fejlesztői környezet szolgáltatta.

```
COVERAGE_REPORT_START
Total source files:          8
Total source lines:          372
Covered lines:               354     (95.2%)
Not covered lines:           18      (4.84%)
Not covered & not marked lines: 0    (0%)
Overall coverage:            372     (100%) <<<<
COVERAGE_REPORT_END
```

5. ábra
A tesztelés eredményének kiértékelése

3. Eredmények

Az ismertetett verifikációs módszereket, a módszereket implementáló tesztelési keretrendszert egy alkalmazásspecifikus operációs rendszer verifikálásánál alkalmaztuk. Az operációs rendszer on-line vasúti irányítórendszerek alkalmazásainak futtatására készült. A tesztelt alkalmazás jelenleg is valós környezetben működik.

A rendszer tesztelése során az iteratív tesztgenerálás módszerét alkalmaztuk minden funkcionális csomag esetén. A tesztelés során kidolgoztuk a teljes rendszerre a öndokumentáló, újra végrehajtható teszteseteket tartalmazó tesztelő rendszert. A tesztelés során teljes forráskódra nézve elértük a száz százalékos utasítás lefedettséget. Az operációs rendszer, a kidolgozott tesztelő rendszer segítségével sikeresen megfelelt a CENELEC szabványban [11] definiált hivatalos auditálási procedúrán.

4. Összefoglalás

A dolgozatban olyan rendszerverifikácós módszereket mutattunk be, melyek lehetőséget adnak nagyméretű és összetett objektumorientált rendszerek tesztelésére. Az ismertetett módszereket tesztelési keretrendszer formájában implementáltuk és sikeresen alkalmaztuk egy alkalmazás-specifikus operációs rendszer tesztelésénél és auditálásánál. A kidolgozott módszerek igen hatékonynak bizonyultak, segítségükkel a tesztek kidolgozására, illetve a tesztelésre fordított idő a legóvatosabb becslések alapján is kevesebb, mint a felére csökkent az eredetileg tervezett, e módszerek alkalmazása nélküli állapothoz képest.

Köszönetnyilvánítás

A tesztelési keretrendszer kialakításában történő közreműködésükért köszönetet mondok dr. Várady Péternek és Asztalos Attilának. A kutatómunkát támogatta az Országos Tudományos Kutatási Alap (OTKA-F046726), valamint a Gazdasági és Közlekedési Minisztérium (AKF-05-0093, AKF-05-0408, RET-04-2004).

Irodalom

[1] D. Ince: Software Testing in J. McDermin (ed.): Software Engineer's Reference Book, Butterworth-Heinemann Ltd., 1991.
[2] J.J. Chilenski, S.P. Miller: Applicability of Modified Condition Decision Coverage to Software Testing, Boeing Company and Rockwell International Co.,1993.
[3] P. Várady: Konzeption und Entwicklung einer Analysebibliothek zum Test des Verhaltens eingebetteter Software, Diploma Thesis in German, FZI-MRT Karlsruhe, 1997.
[4] H. Younessi: Object-Oriented Defect Management of Software, Prentice-Hall, Inc., USA, 2002.
[5] B. Benyó, J. Sziray: The Use of VHDL Models for Design Verification, IEEE European Test Workshop (ETW2000), Cascais, Portugal, May 23-26, 2000, ISBN 0-7695-0701-8
[6] P. Várady, B. Benyó: A systematic method for the behavioural test and analysis of embedded systems, INES 2000, 4th IEEE International Conference on Intelligent Engineering Systems 2000.
[7] IPL Information Processing Ltd: Advanced Coverage Metrics for Object-Oriented Software, White paper of IPL Information Proc. Ltd, 1999.
[8] J. Sziray, B. Benyó., I. Majzik, L. Kalotai, J. Góth, T. Heckenast: Quality Mangement and Verification of Software Systems, Research Report (in Hungarian), (KHVM 47/1998), Budapest, 2000.
[9] M. R. Woodward, D. Hedley, M. A. Hennell: Experience with Path Analysis and Testing of Programs, IEEE Transactions on Software Engineering, Vol. SE-6, No.3, pp.278–286., May 1980.
[10] David E. Avison, Hanifa U. Shah: The Information Systems Development Lifecycle: A First Course in Information Systems, McGraw-Hill Book Company, Great Britain, 1997.
[11] European Standard EN-50128, Final Draft, Railway Applications: Software for Railway Control and Protection Systems, CENELEC: European Committee for Electrotechnical Standardization, 1997.
[12] M. Dorman: C++ "It's Testing, Jim, But Not As We Know It", White paper of IPL Information Proc. Ltd, 1999.
[13] I. Jacobson, G. Booch, J. Rumbaugh: Unified Software Development Process, Addision-Wesley, USA, 1999.

Biometriával ötvözött digitális aláírás

JEGES ERNŐ, HORNÁK ZOLTÁN

BME, Méréstechnika és Információs Rendszerek Tanszék, SEARCH Laboratórium

jeges@mit.bme.hu

hornak@mit.bme.hu

Kulcsszavak: biometria, nyilvános kulcsú kriptográfia, hibajavító kódolás, csatornakódolás

Az elektronikus ügyintézések biztonsága megköveteli, hogy biztonságos elektronikus aláírási technikák álljanak rendelkezésünkre, amelyek erős kriptográfiai módszerek révén biztosítják, hogy a dokumentum aláírója azonosítható, az aláírás ténye letagadhatatlan, valamint az aláírt dokumentum tartalma sértetlen legyen. A bemutatott biometriával ötvözött digitális aláírás technológia alapvetően az aláíró fél azonosításának, és az aláírás letagadhatatlanságának megerősítésére koncentrál.

A jelenlegi elektronikus aláírási rendszerekben a leggyengébb, nem erősíthető láncszemet a valódi személy és az őt azonosító titkos kulcs közötti kapcsolat képezi. A titkos kulcsot tartalmazó eszköz eltulajdonítható, míg a kulcshoz való hozzáférés a mai megoldásokban csupán jelszavas megoldással védhető, ami nem bizonyító erejű.

Az általunk javasolt módszer alapötlete az, hogy az aláíró felet azonosító titkos kulcsot olyan kódolt formában tároljuk, hogy csak a kulcs tulajdonosának ujjnyomatából kiolvasható adat segítségével legyen visszaállítható, és csak így lehessen aláírást készíteni vele. Ilyen módon a kódolt titkos kulcs, vagyis a kártya eltulajdonítása révén sokkal nehezebb visszaélni, hiszen az aláírás elkészítéséhez szükséges még – a jelenlegi PIN kód mellett – a tulajdonos ujjnyomata is. Fontos megjegyezni, hogy ez a módosítás nem befolyásolja a nyilvános kulcs használatát, így a tanúsítvány és az aláírás visszaellenőrzésének folyamata teljesen kompatíbilis a jelenlegi PKI ajánlásokkal és a meglévő alkalmazásokkal.

1. A digitális aláírás

Dokumentumok hitelességének igazolására, illetve vitás esetekben az eredetiség eldöntésére egy rendkívül egyszerű, de mégis kellően hatásos módszer a hagyományos, kézírással készített aláírás. A kézírást pontosan utánozni nehéz, a hamisított aláírást szakértők nagy bizonyossággal képesek felismerni, illetve megkülönböztetni a valódi hiteles aláírástól. Mivel tehát az írás egyértelműen az adott személyre jellemző, a *hamisíthatatlanságon* túl az aláírás egy igen fontos tulajdonsága a *letagadhatatlanság*. Ha valaki valamit kézírásával leír, akkor nem tudja annak tényét letagadni, mert a kézírását nehezen tudja úgy megváltoztatni, hogy az azonosságot szakértők ne tudnák megállapítani.

A hagyományos kézírással történő aláírás analógiájára született meg a digitális aláírás az elektronikus dokumentumok hitelességének az igazolására. A jelenlegi elektronikus aláírások alapja egy titkos kulcs (az elektronikus aláírásról szóló törvény szóhasználatával élve „aláírás létrehozó adat”), amely használatával – feltételezve, hogy az csak a jogosult személy *birtokában* lehet – a háttérben alkalmazott kriptográfia garantálja, hogy az illető nevében más digitális aláírást nem képes készíteni. Ezen rendszerek leggyengébb láncszeme pontosan ez a birtokviszony, tehát a titkos kulcs eltulajdoníthatóságának a kérdése. A vonatkozó törvény úgy rendelkezik, hogy akár a jogos felhasználó, akár más írt alá a kulcs felhasználásával, a kötelezettségvállalás következményeit a kulcs tulajdonosának kell teljesítenie, vagyis nem a valós aláírót, hanem a felelősséget vizsgálja, ami lényeges különbség.

Az aláíró személy és a nyilvános kulcs egymáshoz rendelésének megoldásai közül egyértelműen a *nyilvános kulcsú infrastruktúrát* (PKI), vagy az általa menedzselt elektronikus igazolványok rendszerét támogatják a törvényi szabályozások [1]. A PKI rendszerében egy mindenki által megbízhatónak elfogadott harmadik fél *(Certificate Authority, CA)* elektronikus dokumentumba foglalja az adott személy nevét és más azonosító jellemzőit, illetve a nyilvános kulcsát, majd ezeket együttesen a tanúsító intézet a saját titkos kulcsával aláírja, ezzel biztosítva, hogy észrevétlenül nem történhet változtatás a rögzített adatokban.

Eddig a titkos kulcs birtoklását általában chipkártyával és a hozzá tartozó titkos PIN kóddal oldották meg. Az ennél jóval erősebb biometrikus módszereket szinte kizárólag a PIN kódot kiváltó megoldásként alkalmazták, ami azonban nem gátolja meg, hogy a titkos kulcs illetéktelen személy birtokába kerüljön.

Cikkünkben egy olyan, a hagyományos nyilvános kulcsú infrastruktúrára épülő módszert mutatunk be, ahol a biometrikus azonosítás – esetünkben ujjnyomat alapú azonosítás – olyan módon és olyan mértékben épül be az elektronikus aláírás folyamatába, hogy a titkos kulcs az esetlegesen alkalmazott chipkártya eltulajdonításával és feltörésével sem szerezhető meg, mert a kulcs gyakorlatilag a jogosult személy ujjnyomatában van eltárolva, illetve annak segítségével van kódolva.

2. Biometria bevonása az aláírási folyamatba

Számos módszer ismert a biometria területén belül az ujjnyomat alapú megoldástól a hang-, illetve arcfelismerésen keresztül a retina és írisz vizsgálatáig. Az egyes módszerek jelentős eltérést mutatnak abból a szempontból, hogy mekkora bizonyossággal, milyen tévedési arányokkal képesek felismerni személyeket, illetve mennyire könnyű őket becsapni (ujjnyomatról készült szilikon replikával, egy hangfelvétel viszszajátszásával vagy éppen egy fénykép felmutatásával). Ennek tükrében az olcsó és egyszerű, illetve a drága de megbízhatóbb megoldások között lehet válogatni; biometrikus digitális aláírás megvalósításához legcélszerűbbnek az ujjnyomat alapú megoldás mutatkozott, amely egyrészt optimális az ár-teljesítmény viszony tekintetében, másrészt az ujj leolvasóra helyezése emberi szempontból is jól kifejezi az aláírás folyamatát és szándékát.

Az ujjnyomatok azonosítására számos alapmódszert dolgoztak ki az elmúlt évtizedek, sőt évszázadok során, amióta rájöttek, hogy nyomozati és bizonyítási eljárásokban is sikerrel alkalmazhatják az ujjnyomat egyediségét. Az ujjnyomatok gépi kezelése során ezen daktiloszkópiai módszerek közül a minutia alapú azonosítási eljárás vált szinte egyedülállóvá.

1. ábra Példa ujjnyomat képek

A fodorszálak végét, különféle elágazásaik illetve összefutásaik helyét nevezzük minutia-pontoknak. Ezen pontok elhelyezkedése rendkívül jellemző egy adott ujjra. A legtöbb azonosítási módszer kizárólag ezen pontok relatív elhelyezkedéséből dönti el két ujj azonosságát vagy különbözőségét.

A minutia pontok esetében a következő jellemzők vizsgálhatóak:

– *Elhelyezkedés:* a pont síkbeli koordinátája, mely megadható egy képen belül abszolút vagy egy alapponthoz viszonyított relatív értékkel.
– *Irányultság:* minden ponthoz rendelhető egy irányszög, amit az érintett fodorszál vagy fodorszálak iránya határoz meg.
– *Görbület:* a barázda irányultság megváltozásának mértéke.

Az ujjnyomat-képet az ujjnyomat-olvasó eszköz segítségével tapogathatjuk le, amely általában élőujj- és ujjnyomat-replika detektálást is tartalmaz, tehát meg tudja állapítani, hogy az olvasott ujj ténylegesen élő ujjról származik, vagy csak egy, az adott ujj redőit utánzó, úgynevezett replika került az olvasóra.

A minutia alapú ujjnyomat azonosítási módszer alapja tehát a fodorszálak meghatározása, majd azok alapján a minutia pontok helyzetének a vizsgálata. A módszerek által meghatározott pontokat általában különféle szűrőknek vetik alá, kihasználva, hogy a valódi minutia pontok – akárcsak a valódi fodorszálak – jellemzői bizonyos szabályokat követnek. A fedésbe hozható pontok vizsgálatával megállapíthatjuk két ember ujjnyomatának azonosságát, ezáltal azonosítva az ujjnyomatokhoz tartozó embereket.

A biometriával ötvözött, titkos kulcsot tárolni képes rendszerek zöme, mint már említettük, egyszerűen biometrikus módszerekkel védi a hozzáférést a letárolt titkos kulcshoz. Ezzel szemben mi azt a célt tűztük magunk elé, hogy a titkos kulcsot egyáltalán nem tároljuk, hanem azt a kulcspár generálást követően kódoljuk, majd töröljük. A későbbiekben, ha a titkos (aláíró) kulcsra van szükség, a kódolt információból a titkos kulcs visszaállítása csak az ujjnyomatkép ismeretében lehetséges.

A digitális aláíráshoz szükséges titkos kulcs tárolásának megvalósításához a minutia pontok fent felsorolt jellemzői azért fontosak, mert minden egyes független jellemző felhasználható a titkos kulcs bitjeinek a kódolására az ujjnyomat-képben.

3. A titkos kulcs eltárolása az ujjnyomatban

Elektronikus aláírás készítéséhez valamilyen kriptográfiai szempontból erős rejtjelkulcsra, bináris adatra van szükség. Elviekben ez a bitsorozat magából az ujjnyomat-képből is származtatható lenne, de mivel azonos ujjnyomatokból minden esetben bitről bitre azonos bináris adatot kell kapnunk, az olvasó bizonytalansága és az eltérő olvasási környezetből fakadó különbségek miatt ez így nehezen megvalósítható módszernek bizonyult.

A megvalósított módszerünk lényege ennek megfelelően az, hogy a regisztráció során a kinyert minutia pontok alapján egy próba pontsorozatot *(challenge minutia vektort)* tárolunk el, amelybe egyrészt a valós pontokon túl álpontokat is belekeverünk, másrészt a pontok irányultságát megváltoztatjuk – ezáltal kódolva azt az információt, ami végső soron a titkos kulcs visszaállíthatóságát biztosítja. A visszaállítási bizonytalanságok természetesen ilyen módon is megmaradnak, ezeket azonban már képesek vagyunk kezelni hibajavító kódolás alkalmazásával.

A regisztráció során a titkos kulcs generálásához felhasznált (eredeti) bitsorozatot ezek után aláíráskor úgy rekonstruáljuk, hogy a fenti challenge minutia vektort az aktuálisan levett ujjnyomat-mintából kapott minutia pontokkal „ütköztetjük”. Ezen művelet eredményeképpen, a megfelelő hibajavítás után visszakapjuk az eredeti bitsorozatot, amely segítségével képesek vagyunk a tanúsítványban szereplő publikus kulcshoz tartozó titkos kulcs újragenerálására.

A fentiekből már látható, hogy a módszer két fő folyamatból áll: a regisztrációból és az aláírásból. Az első során áll elő a publikus kulcsot tartalmazó tanúsítvány, illetve a challenge minutia vektor; az aláírási folyamat során a challenge minutia vektor és a levett minutia pontok alapján újra előáll a titkos kulcs, amely segítségével megtörténik az aláírás.

A regisztráció lépései a következők (a vastag betűk a folyamat során előálló, elmentett eredményeket jelölik):

- Megfelelő hosszú véletlen bináris vektor előállítása.
- A bináris vektor bővítése megfelelő számú paritásbittel a hibajavításhoz.
- Az így előállt teljes bitsorozat, illetve a regisztrált ujjnyomat-mintából nyert minutia pontok alapján a **challenge minutia vektor** előállítása, elmentése. A kódolás menetét alább mutatjuk be.
- A teljes bitsorozat alapján RSA kulcspár generálása, a titkos kulcs törlése.
- **Tanúsítvány** készítése a publikus kulcs alapján, kapcsolódva valamely létező nyilvános kulcsú infrastruktúrához.

Az aláírás során a feladat a titkos kulcs rekonstruálása. Ehhez az aktuálisan levett ujjnyomat mintán kívül a rendelkezésünkre áll a challenge minutia vektor, illetve a létrejött titkos kulcs ellenőrzéséhez a neki megfelelő, tanúsítványba foglalt publikus kulcs.

- A teljes bitsorozat visszaállítása a challenge minutia vektor és az aktuális ujjnyomat minutia pontjai alapján. Ez a regisztrációkor használt kódolás inverz műveleteként is felfogható, tehát mint egy dekódolás.
- A teljes visszaállított bitsorozat hibajavítása.
- A teljes bitsorozat alapján az RSA kulcspár generálása.
- A generált publikus kulcs és a tanúsítványba foglalt publikus kulcs összehasonlítása. Egyezés esetén a titkos kulcs elfogadása, aláírás; ellenkező esetben hibajelzés.

A továbbiakban ismertetjük a fenti rendszer megvalósítása közben felmerült problémákat, illetve az ezen problémákra általunk adott megoldásokat.

Kódolás és dekódolás

A kódolás folyamata nem más, mint a challenge minutia vektor előállítása. Ennek során a teljes bitsorozat bitjeinek megfelelően generáljuk a challenge vektor pontjait. Esetünkben a bitsorozatot ötösével kezeljük, amiből nyilvánvaló, hogy a hibajavító kódokkal bővített teljes bitsorozat hosszának öttel oszthatónak kell lennie. Az egyes bit-ötösöket a következő séma szerint kódoljuk:

0	1	2	3	4
Ál / valós	Irányultság módosítása			

2. ábra Az egy minutia-pontnak megfelelő ötbites futam

Mint az az ábrán látható, a teljes bitsorozatot ötös futamokra bontjuk, és ezen részek alapján generáljuk sorra a challenge minutia vektor pontjait.

Ha egy ilyen futam első (0.) bitjének értéke 1, valós minutia pontot választunk a regisztráltak közül, ellenkező esetben véletlenszerű ál-minutia pontot generálunk a pont alatt található fodorszálak irányultságával. A további 4 bit értékétől függően az adott ál- vagy valós pont szögét módosítjuk *dFi* szöggel a következő táblázat szerint:

Kód bitek	dFi (fok)	Kód bitek	dFi (fok)
0000	0,00	1100	90,00
0001	11,25	1101	101,25
0011	22,50	1111	112,50
0010	33,75	1110	123,75
0110	45,00	1010	135,00
0111	56,25	1011	146,25
0101	67,50	1001	157,50
0100	78,75	1000	168,75

3. ábra A 180 fokos szögkódolás

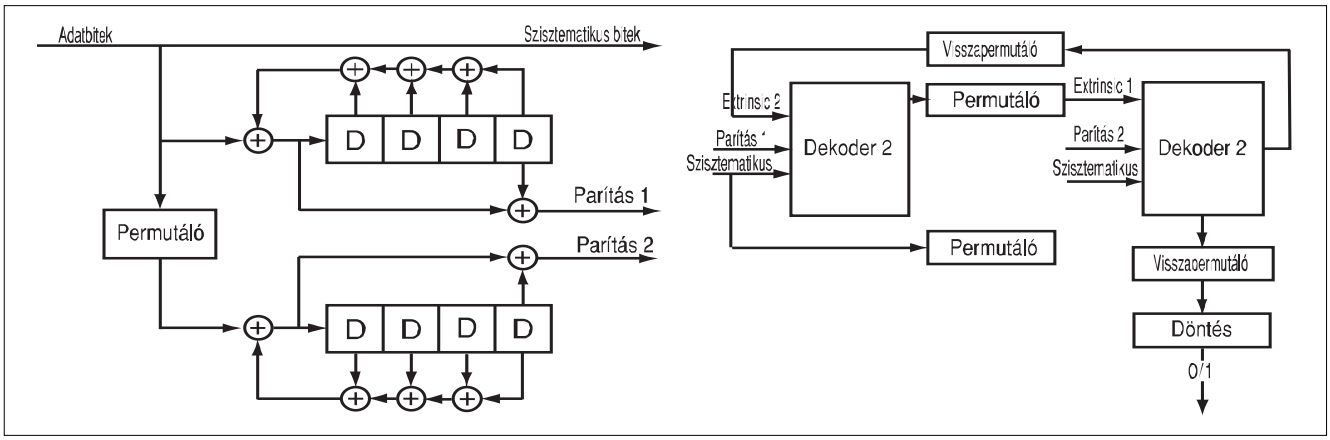
A fenti táblázatból kiolvashatjuk, hogy amennyiben az 1-4 bitek értéke *1101*, a vektorban szereplő minutiapont eredeti szögét 101.25 fokkal (9 x 11.25°) kell módosítani (modulo 180). Ez tulajdonképpen megfelel a 11.25 fokos lépésközben felírt szögmódosítás értékek Gray-kódolásának, amely tulajdonsága az, hogy a szomszédos értékek Hamming távolsága 1. Ez esetünkben azért fontos, mert ez a kódolás a szögmegállapítás bizonytalanságából adódó bithibákat a minimálisra csökkenti, ráadásul a következőkben ismertetett hibajavító kódoláshoz illeszkedve dekódoláskor a határhelyzetben talált szögek esetében a megfelelő bitekben törléses hibákat jelezhetünk, ezzel is növelve a hibajavító képességet.

Hatékony hibajavítás

Az általában hibajavításra használt csatornakódolókkal ellentétben esetünkben a kódolás és a dekódolás ideje kevésbé volt kritikus paraméter, a hibajavító módszer kiválasztását inkább az a tény határozta meg, hogy az ujjnyomat általunk választott kódolása jelentős bizonytalanságot hordozott. Ennek megfelelően – a szokásos fogalmakkal élve – egy igen alacsony jel/zaj viszonnyal rendelkező csatornával álltunk szemben.

Számos alternatíva megvizsgálása után nyilvánvalóvá vált, hogy például az űrtávközlésben használt, hatékony hibajavító paraméterekkel rendelkező, és jól skálázható Turbó-kódolás az, amely igényeinket kielégítheti. A Turbó-kódoló alapötlete az, hogy két (vagy akár több), párhuzamosan kapcsolt rekurzív konvolúciós kódolót használunk [3].

Az első az eredeti szisztematikus bitekből, a második pedig azok permutációjából állít elő paritásbiteket. Az ilyen módon párhuzamosan képzett paritásbiteket a szisztematikus bitekhez fűzzük; így, amennyiben az egyenként előállt paritásbitek száma megegyezik a szisztematikus bitek számával, egy 1/3 jelsebességű kódot kapunk (a kódszó hoszsza a szisztematikus bitek



4. ábra Két konvolúciós kódolót tartalmazó Turbó-kódoló (balra) és dekódoló (jobbra)

száma plusz a két kódoló által előállított paritás bitek száma). További ötlet, amely a Turbó-kódolást igen jól skálázhatóvá teszi az, hogy nem viszünk át a csatornán minden paritásbitet, hanem közülük bizonyos minta szerint törlünk. Így tetszőleges jelsebességet érhetünk el, természetesen valamelyest veszítve a hibajavítási képességen.

A konvolúciós kódok dekódolásához hasonlóan dekódoláskor itt is a kapott bitek alapján becsüljük az egyes kódolási lépésekben a konvolúciós kódoló állapotát, ezáltal az eredeti szisztematikus bitek értékét is, felhasználva a csatorna kimenetén vett értékeket és az előző dekódoló által szolgáltatott információt. Egy tipikus, két konvolúciós kódolót tartalmazó Turbó-kódoló és az ennek megfelelő dekódoló elrendezés a *4. ábrán* látható.

Az előzőekben bemutatott kódolásnak megfelelően az egyes bitek átvitelének modellezésére létrehoztuk a nem-szimmetrikus bináris törléses csatorna-modellt (NBEC). Ez a modell az egyes bitek átvitelekor értelmezi a törléses hibát, így különböző hibaparaméterrel írhatjuk le mind az egyszerű, mind a törléses hiba valószínűségét különböző bemeneti bit-értékek esetében (innen az *aszimmetrikus* elnevezés). Az NBEC csatornát tehát négy hibaparaméterrel (p_{0X}, p_{01}, p_{1X} és p_{10}) írhatjuk le.

Mivel a kódolás bit-ötösöket rendel egy-egy minutiaponthoz, és ezt az öt bitet különböző szabályok szerint kódoljuk, az egyes bitek (0-4) esetében más-más hibaparaméter-négyest definiálhatunk. Ilyen módon kapjuk meg a tényleges alkalmazott $NBEC_5$ csatornamodellt, amelyet 5, egymástól független NBEC csatornaként képzelhetünk el *(5. ábra)*.

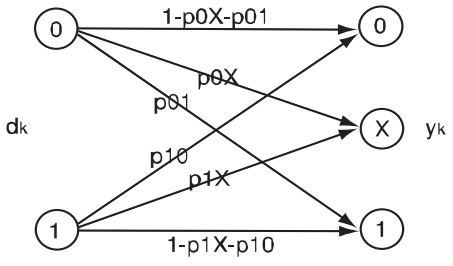
Mint már említettük, a paritásbitekkel kiegészített teljes bitsorozat hosszának oszthatónak kell lennie ötel. Az átvitelre nem kerülő paritásbitek adott minta szerinti törlése miatt 8-al is osztható hosszú bitsorozatot kell választanunk. A különböző lehetséges paritásbit hosszak áttekintése után a 120+120 bites kódszó mellett döntöttünk, ami azt jelenti, hogy a létrejött paritásbitek felét töröljük, ezáltal egy 1/2 jelsebességet kódolást kapunk. Ilyen módon a 120 darab szisztematikus véletlenbit kriptográfiai értelemben erős kulcsot biztosít, miközben a challenge vektor 240/5=48 darab minutia-pontot tartalmaz.

Az ötös futamokon belül a 0. bit kódolását figyelembe véve tehát átlagosan 24 valós minutiára lesz szükségünk, hiszen a véletlen bitek átlagosan fele nulla, ami megfelel az egy ujjnyomatban fellelhető minutiapontok eloszlásának, hiszen a 600 ujjat tartalmazó adatbázisunkban az átlagos minutia-pont szám 40-re adódott.

Determinisztikus kulcspár generálás

Ahhoz, hogy a korrektül visszaállított bitsorozatból minden esetben ugyanazt a kulcspárt kapjuk, módosítanunk kellett az OpenSSL véletlenszám-generátorát, amelyre a kulcsgenerálás támaszkodik. Így az általunk

5. ábra A nem szimmetrikus bináris törléses csatorna (NBEC), és az NBEC5 statisztikailag meghatározott hibaparaméterei



	0	1	2	3	4
p_{00}	0,53	0,46	0,45	0,42	0,36
p_{0X}	0,00	0,02	0,02	0,03	0,05
p_{01}	0,01	0,02	0,02	0,04	0,09
p_{10}	0,13	0,02	0,02	0,04	0,08
p_{1X}	0,03	0,02	0,02	0,03	0,05
p_{11}	0,31	0,45	0,46	0,42	0,38

előállított bitsorozatot adagolva a generátor magjának, az OpenSSL minden esetben azonos véletlenszámsorozatot, ezáltal azonos kulcspárt generál.

4. Összefoglaló

Mint a biometrikus rendszerek esetében általában, a biometrikus digitális aláírás esetében is a téves visszautasítások *(FRR – False Rejection Rate)* és a téves elfogadások *(FAR – False Acceptance Rate)* száma az alapvető hibaparaméter. Esetünkben az első paraméter elsősorban a képvételi hibák, a mintavétel bizonytalanságából adódik, a másodikat pedig elsősorban az erős hibajavítás növelheti. Egy biometrikus rendszer hangolásánál a két érték általában egymás ellen hat, így a projektünk során is meg kellett találnunk a paraméterek azon halmazát, ahol a két hibaparaméter a megfelelő értéket mutatja.

Az általunk választott paraméterek mellett, tesztjeink során az FAR 10^{-6}-nál kisebb értékre adódott, míg az FRR értéke 15% körüli volt. Mindkét érték a biometrikus rendszerek esetében elfogadható határon belül van, eme utóbbi azonban vélhetően tovább csökkenthető különböző szűrők alkalmazásával, illetve a mintán fellépő nem-lineáris torzulások megfelelő kezelésével.

Összefoglalásként elmondhatjuk, hogy a biometrikus digitális aláírás megvalósítható, és több szempontból is erősebb védelmet nyújt, mint a mára már hagyományossá váló pusztán chipkártyán alapuló titkos kulcs tárolási módszerek. A kidolgozott eljárás rendkívüli erénye, hogy mind a tanúsítvány formátuma és tartalma, mind az aláírás ellenőrzés folyamata teljesen megegyezik a mai megoldásokkal, teljes mértékben kompatibilis azokkal.

A bemutatott eljárás a digitális aláírás használatának széleskörű elterjedése által gerjesztett növekvő felhasználói igények mellett komoly sikerre számíthat.

Köszönetnyilvánítás

A kutatási projektet az Infokommunikációs Technológiák és Alkamazások támogatta (IKTA-00160/2002).

Irodalom

[1] 2001. évi XXXV. törvény az elektronikus aláírásról; www.ihm.hu/miniszterium/jogszabalyok/ealairas.pdf
[2] The OpenSSL Project; OpenSSL: The Open Source toolkit for SSL/TLS; www.openssl.org/
[3] Guangchong Zhu, Performance Evaluation of Turbo Codes. Queen's University, Kingston, Ontario, Canada, 1998. http://markov.mast.queensu.ca/Papers/zhu_proj98.ps

A testlengés és a kéz tremor méréstechnikája

BRETZ KÁROLY JÁNOS

BME, Villamosmérnöki és Informatikai Kar, Méréstechnika és Információs Rendszerek Tanszék
bretz@mit.bme.hu

Reviewed

Kulcsszavak: stabilometria, tremor, stressz, méréstechnika

A testtartás stabilitását bonyolult bio-szabályozási rendszer tartja fenn. A szabályozott jellemző a test tömegközéppont függőleges vetületének pozíciója. Amennyiben ez a pont a bázisfelületen belül van, az állás stabil [2,3,6]. Nagyszámú kísérlet bizonyítja, hogy a rendszer az optimális pozíciót keresi, miáltal a tömegközéppont függőleges vetülete a bázisfelület közepe felé tart. A szabályozás pontossága egyénenként jelentős mértékben eltérő. Állásban a test lengéseket végez. A visszacsatolás három, funkcionálisan jól elkülöníthető körben történik, nevezetesen: a vizuális, vesztibuláris és a proprioceptiv „visszacsatoló rendszerek" részvételével. A stabilometriában a fentiekben definiált szabályozott jellemzőt közvetve regisztráljuk, a nyomásközépponti trajektóriák meghatározásával.

1. Bevezetés

Az állás stabilitásának a jelentősége nagy, bár ennek értelmét csak eleséskor észleljük. Ipari példaként az építőállványon végzett munkát említjük [3].

A tremor valamely testrész akaratlan, ritmikus remegése [6] Létrejöttét az antagonista izmok reciprok innervációjával magyarázzák. A tremor a kéztartás instabilitásának és a mozgási rendellenességeknek leggyakoribb tünete.

A kar, a kéz és az ujjak tremorjának diagnosztizálása a klinikumban történik. Egészséges embernél, a munkaköri alkalmasság eldöntésénél, a finommechanika és optika, valamint a mikroelektronika egyes területein vélelmezik a fiziológiás tremor vizsgálatának fontosságát. A kéz és az ujjak tremorja akadályozó tényező lehet a mikromanipulációk esetén, kisméretű szerszámok használatánál [6].

A tremor típusainak megkülönböztetéséhez Fourier-analízissel meghatározható domináns frekvencia ad felvilágosítást [6,9,10]. Ennek alapján:

3-4 Hz	kisagyi eredetű tremor
4-6 Hz	esszenciális (idősebb korban), Parkinson-, izomtónus- és pszichés eredetű tremor.
6-12 Hz	esszenciális (fiatal korban), fiziológiás, álló helyzetben jelentkező, izomtónus- és pszichés eredetű tremor.

Selye (1953) megfogalmazásában a stressz „nem specifikus válasz, amely különböző specifikus megnyilvánulásokra szuperponálódik". Az állás stabilitását, a tremor paramétereit befolyásoló hatások egyike a stressz lehet.

Jelen munka célja a testtartás stabilitásának, a kéz tremornak és a stressz faktornak meghatározására fejlesztett, illetve felhasznált méréstechnikai eljárások ismertetése és e három pszichofiziológiai paraméter kölcsönhatásainak bevezető tanulmányozása.

2. Metodika

A kísérletekben egyetemi hallgatók vettek részt. Tizenkilenc személy adatait értékeltük.

A mérések első részénél az alanyok Romberg-kísérleti pozícióban egyenesen álltak, zárt lábbal, előrenyújtott kézzel, a tenyerüket lefelé fordítva, nyitott és csukott szemmel, 20-20 másodpercig [3]. Ebben a testhelyzetben az egyensúlyi szabályozást és a kéz tremort mértük *(1. ábra)*.

A kísérletek második szakaszában, a tremor mérésénél, előrenyújtott kézzel ültek a résztvevők. A mérési idő 20 másodperc volt.

A harmadik részben a manualitást és a tartás biztonságát ellenőriztük az e célra tervezett eszközzel, ülőhelyzetben, könyöktámasszal, 30 másodperces mérési idővel [6].

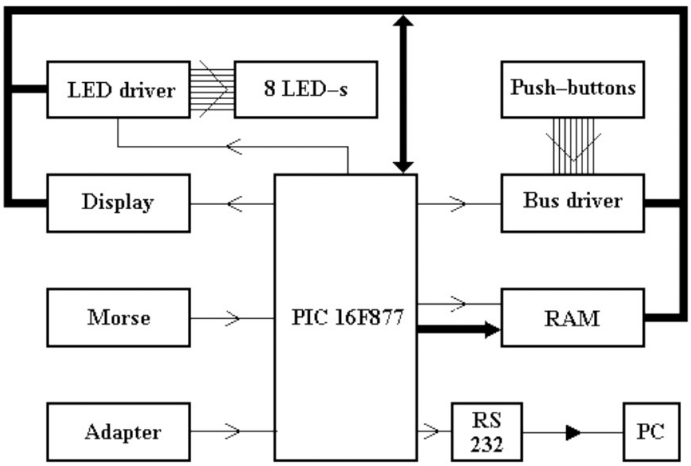
Az irodalom az állás egyensúlyi stabilitását és a tremort modellek felállításával is tárgyalja [5,7,13]. Az előbbi vizsgálatának legegyszerűbb módja az, hogy a testet egyetlen merev tömegnek tekintik és az invertált inga modellt alkalmazzák.

Megvizsgálták, hogyan függ az izületekben ható forgatónyomatéktól a testszegmens gyorsulása [5,13].

Az invertált inga egyensúlyozásának vizsgálatára PD szabályozó modell is eredményre vezet. A szabályozás késleltetésének kritikus értékére kiszámított adat egybeesik a stabilometriával nyert eredményekkel.

3. Mérőeszközök

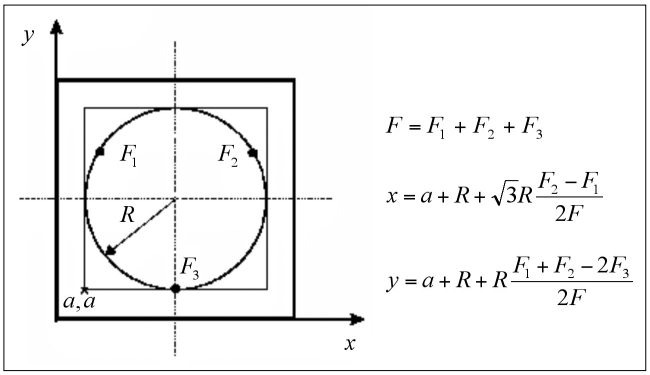
A stabilométer berendezés három érzékelővel ellátott erőmérő platformot (platformokat), hatcsatornás erősítőt, mikroszámítógépet – utóbbit analóg multiplexerrel, A/D-val, mikrokontrollerrel és interfész áramkörrel –, valamint egy személyi számítógépet tartalmaz *(2. ábra)* [3].



1. ábra
Romberg-teszt stabilométeren és finommechanikai ipari munkaalkalmasságot tesztelő, tremor mérésére szolgáló berendezés blokksémája

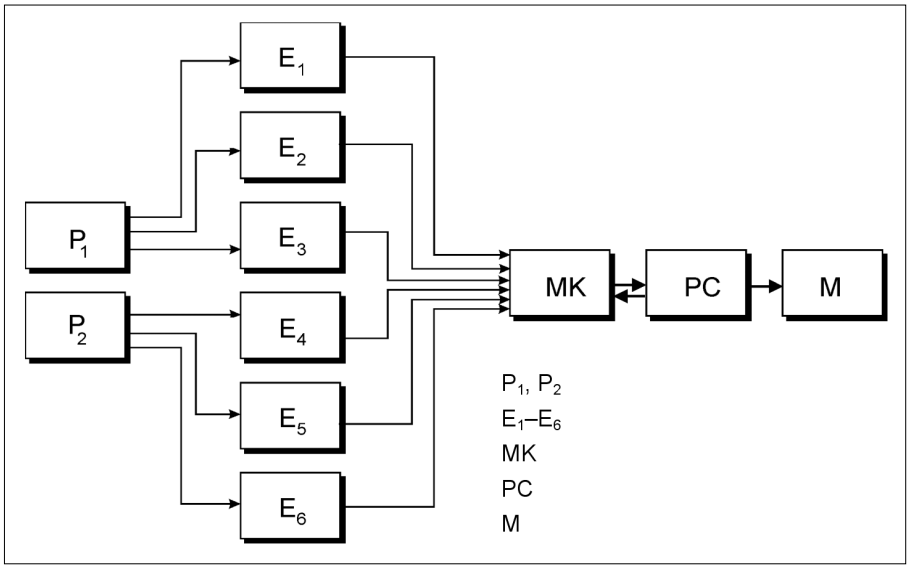
A platform fedőlapjának mérete: 0,5x0,5 m *(3. ábra)*. A mérőrendszer linearitása +1,5%, hiszterézise +1,5%. A nyomásközéppont x-y koordinátái 1 mm felbontással adottak. A mintavételi frekvencia állítható a 20-1000 Hz tartományban. A nyomásközéppont mozgásának trajektóriáit: a stabilogramot, ennek idődiagramját a frontális és a szagittális irányú felbontásban, ez utóbbiak Fourier-spektrumát, valamint a stabilogram útvonal hosszát regisztráljuk [3].

A stabilométerrel összekapcsolt, azzal szinkron működő mérőkészülékünk a tremort regisztrálja [6]. Ennek részei középső ujj végéhez erősített kétdimenziós gyorsulásmérő, DC erősítők, A/D és interfész. A gyorsulásmérő érzékelő az Analog Devices Co. ADXL202 típusú eszközét tartalmazó áramkör. Mérési tartománya: ±2 g.



3. ábra A stabilométer elvi felépítése és a stabilogram előállításának egyenletei

2. ábra A stabilométer blokksémája



Ipari alkalmasság-vizsgálati célra kifejlesztett készülékünk a tremort és a reakcióidőt méri. Az elektronikus egységét PIC16F877 mikrokontroller vezérli. Ezen kívül fogadó adaptert, valamint az órás-csavarhúzó és a miniatűr forrasztópáka modelljeit tartalmazza (1. ábra).

A stressz és a kardiovaszkuláris állapot felmérésére Cardioscan készüléket használtunk (Energy Laboratory Technology GmbH, Germany).

Ez a készülék a stressz szintjét 0-100%-os skálán, a kardiovaszkuláris rendszer állapotát 1-5 pontos skálán határozza meg és az EKG-jel numerikus analízisét szolgáltatja. A mérési idő 1 perc.

Összehasonlító adatsorként szolgáltak a Spielberger kérdőívvel analizált „szorongás”, „harag”, „kíváncsiság” és „depresszió” paraméterek [4].

4. Eredmények és értékelés

Az állás stabilitásának méréssel meghatározott egyik jellemzője a karakterisztikus kör sugara: „r”. A karakterisztikus kör a stabilogram mintavételezett pontjainak 68%-át, illetve a 95%-át foglalja magába [3]. Bármelyik használható, de összehasonlítás esetén ugyan azt a mértéket kell használni. Rendszeresen sportoló egyetemi hallgatók adatainak értéktartománya r: 5-8 mm nyitott szemmel, r: 6-10 mm csukott szemmel a 68%-os értelmezéssel.

Jelen vizsgálatban a tremor regisztrátumok többségét fiziológiás amplitúdó és frekvenciatartományba tartozónak lehetett minősíteni. A tartásos kéztremor és a testtartás instabilitása közötti korreláció csak olyan mértékben mutatkozott, amennyiben a merev kartartás következtében a test tömegközéppontjának lengése a karokra is áttevődött. Mivel az ujjak reciprok innervációja lényegesen kisebb időállandójú, mint a testtartásban résztvevő bio-szabályozási rendszeré, ezért az ujjakon mérhető tremor magasabb frekvenciasávba esett, nevezetesen 10-12 Hz-es tartományba.

Megvizsgáltuk a nyomásközéppont mozgásának, mely a Romberg tesztben elfogadható közelítéssel a tömegközéppont mozgására jellemző, és a kéz- (illetve ujj-) tremornak a kapcsolatát. Spektrális vizsgálatok és korrelációs elemzések rávilágítanak arra, hogy jelentősen eltérő időállandójú folyamatokról van szó. A tömegközéppont mozgása lényegesen lassúbb, és kisebb frekvenciájú mint a kéz, illetve az ujj tremorja. Emiatt a két folyamat nem korrelál: r = -0,0186. A merev kartartás azt eredményezi, hogy a kar mozgása követi a test tömegközéppontjának mozgását. A kéz, illetve az ujjak tremorja, oszcillációja erre szuperponálódik. A spektrumvonal ennek megfelelően a kéz (csukló) mozgására nézve 2 Hz körüli helyi maximumot jelez, az ujjak remegésére vonatkozóan kb. 6 Hz-nél mutat maximumot *(4. ábra)*. Utóbbi érték esszenciális (ismeretlen eredetű) tremor meglétére utal.

Elvégeztük a stabilogram és tremor regisztrátum szakaszok korrelációs vizsgálatait, melyekben a szinkronizálás céljából intenzív karmozdulatot tettünk.

Szignifikáns kapcsolatot találtunk a gyorsulásdiagram és a vertikális erő változásai között (kéz-, karmozdulat) A korreláció mértéke r = 0,909 volt.

Karmozdulattal gerjesztett regisztrátum szakaszon a tömegközéppont szagittális (előre-hátra) és frontális (balra-jobbra) mozgásának, valamint az ujj gyorsulásának (gyorsulásmérő) korrelációját is megvizsgáltuk. Mindkét esetben szignifikáns eredményt kaptunk.

A Cardioscan készülék a kardiológiában ismert HRV-t (heart rate variability) és a percenkénti pulzusszám átlagát, szórását, a spektrum jellemzőit értékeli. Az alanyok egyik csoportjánál (tíz fő) írásbeli vizsga előtt a stressz faktor 24%, vizsga után, eredményhirdetés előtt 22% volt. Tehát a feszültség fennmaradt. A másik csoportnál (9 fő) és másik vizsgatárgynál a vizsga előtti átlag 30,2%, vizsga és eredményhirdetés után 17,5%. A stressz faktor skálája 0-100% terjedelmű [4,12].

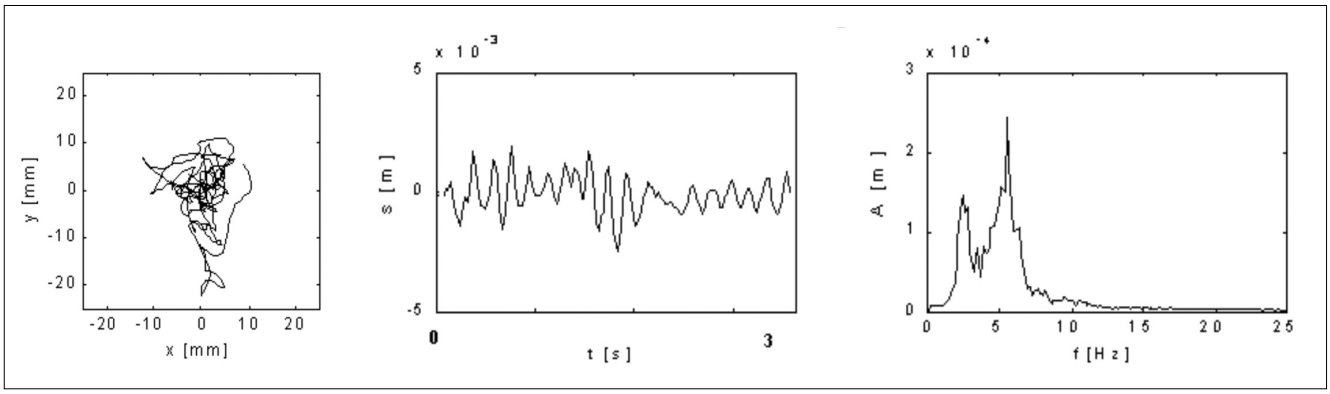
Néhány esetben a stressz nagyobb értékeihez megnövekedett amplitúdójú tremor tartozott. A korrelációs együttható értéke r = 0,32 (n = 15) nem érte el a szignifikancia szintet egészséges egyetemi hallgatók esetében. Megjegyezzük, korábbi vizsgálatunkban pszichiátriai pácienseknél szorosabb kapcsolatot regisztráltunk.

5. Konklúzió

A tremor és a testlengések vizsgálatánál hasonló, vagy azonos modellek állíthatók fel és a transducerek kivételével a mérési eljárások azonosak lehetnek. Egyes esetekben, mint az ipari munkaalkalmasság tesztelésénél, speciális adapterekre lehet szükség. A tartásos kéztremor szuperponálódik a testlengésekre. Az eltérő időállandók miatt ezek nem korrelálnak. Intenzív karmozdulathoz tartozó regisztrátumok között szoros korreláció áll fenn.

A kísérletek szerint a stressz, a szorongás fokozza a tremort. Az egyensúly szabályozási hibáját is növelheti. A jelenség nem általános. Jelen munkában a fenti két paraméter között szignifikáns korrelációt nem regisztráltunk.

4. ábra A stabilogram (balra) és a gyorsulásmérővel regisztrált esszenciális tremor (középen), melyből kétszeres integrálással és a Fourier spektrum meghatározásával (jobbra) diagnosztizálható a tremor.



Köszönetnyilvánítás

Köszönetemet fejezem ki dr. Jobbágy Ákos egyetemi docensnek és dr. Sipos Kornél professzornak a hasznos tanácsaikért. Ez a munka az OTKA T 049357 pályázat támogatásával készült.

Irodalom

[1] Agasin, F.K.: Law of statistical biomechanics. J. Mech. Polymers. Nr.5, Biomechanics. Riga, 1975. pp.590–596.
[2] Allum, J.H.J.: Posturography Systems: Current measurement concepts and possible improvements. In Disorders of Posture and Gait. Eds.: Brandt, T., Paulus, W., Bles, W., Dietrich, M., Krafczyk, Straube, A. Georg, Thieme Verlag, Stuttgart, New York. 1990, pp.16–28.
[3] Bretz, K.: The stability of the human body's equilibrium. Avtoreferat, VAK, Kiev. 1997, pp.1–50. (in Russian)
[4] Bretz, K.J., Sipos, K.: Tremor and stress during college examination. Kalokagathia, 41 (1): 2003, pp.111–115.
[5] Bretz É., Kocsis L, Bretz K.: Balance investigation based on inverted pendulum model of standing human body. In Proc. of the 1st Hungarian Conference on Biomechanics, June 11-12., 2004. pp.43–49.
[6] Bretz, K.J., Lénárt, Á., Bretz, K., Sipos, K.: Investigation of the upper limb tremor and the stability of the human body's equilibrium. In Proceedings of the First Hungarian Conference on Biomechanics, June 11-12., 2004. pp.50–58.
[7] Collins, J.J., De Luca, C.J., Burrows, A., Lipsitz, L.A.: Age-related changes in open-loop and closed-loop postural control mechanisms. Exp. Brain Res. 104., 1995. pp.480–492.
[8] Edwards, R., Beuter, A.: Indexes for identification of abnormal tremor using computer tremor evaluation systems. IEEE Transactions on Biomed. Engineering. 46:1999. pp.895–898.
[9] Jobbágy Á., Fumée, E.H., Harczos, P., Tarczy, M., Krekule, I., Komjáthi, L.: Analysis of movement patterns aids the early detection of Parkinson's disease. Proceedings of the 19th International Conference, Chicago, III, 30.10.-02.11.1997., Washington DC. Institut of Electrical and Electronics Engineers, 1997. pp.1760–1763.
[10] Jobbágy, Á., Harcos, P., Károly, R., Fazekas, G.: Analysis of the Finger-Tapping Test. Journal of Neuroscience Methods, January 30, 2005. Vol. 141/1., pp.29–39.
[11] Kuznyecov, V.V.: The structure of controlled movements of biomechanical limbs. In Morecki, A., Fidelus, K., Kedzior, K., Vit, A. (Eds.): Biomechanics VII-A., University Park Press and PWN – Polish Scientific Publishers, 1993. pp.420–426.
[12] Sipos, K., Sipos, M., Spielberger, C.D.: First results with the Hungarian test anxiety inventory. In Spielberger, C.D., Diaz-Guerrero, R. (Eds.): Cross-cultural anxiety. Hemisphere Publishing Co., Washington D.C., 3:1986. pp.37–44.

Hírek

NewsText

Foltok detektálása mammogrammokon textúra-analízis segítségével

TÓTH NORBERT
Konzulens: dr. Pataki Béla
BME, Villamosmérnöki és Informatikai Kar, Méréstechnika és Információs Rendszerek Tanszék
ntoth@mit.bme.hu

Reviewed

Kulcsszavak: orvosi képfeldolgozás, számítógéppel segített diagnózis, textúra-analízis, döntési fa

Jelenleg a mammográfia az egyik legmegbízhatóbb módszer az emlőrák detektálására. Egy olyan rendszer, amely előzetesen feldolgozná a felvételeket – kiszűrné azokat, amik biztosan negatívak, és felhívná a figyelmet azokra, amelyek gyanúsak – nagyon hasznos lenne. Egy ilyen orvosi döntéstámogató rendszer (ODR) fejlesztése folyik a Budapesti Műszaki Egyetem Méréstechnika és Információs Rendszerek Tanszékén együttműködésben Semmelweis Orvostudományi Egyem radiológus szakembereivel.

1. Bevezetés

A nők daganatos megbetegedései között az egyik leggyakoribb az emlőrák [1]. Statisztika szerint minden nyolcadik nőben élete során kifejlődik ez a betegség. Mivel az emlőrák oka mindmáig ismeretlen, a korai felismerés döntő fontosságú. Korai felismerés esetén az öt éves túlélés esélye 95% körüli.

A mammográfiás szűrés során mindkét emlőről két-két röntgenkép készül, egy oldal-, egy felülnézetből. A mammográfiás képeken a betegség okozta két legjellegzetesebb elváltozás az úgynevezett mikrokalcifikáció, amely apró élesebb kontúrú, fényes pöttyként jelentkezik, illetve a tumorok okozta röntgenárnyék, ami nagyobb, de elmosódottabb foltként jelenik meg.

A mammográfiás szűrésen készített felvételeket két független orvosnak kell diagnosztizálnia, de ez számítógépes segítséggel is történhetne. Egy mammográfiás szűrés óriási mennyiségű felvételt jelent (több százezer nőről készült négy-négy felvétel kiértékelése évente). Minden egyes felvétel értékelése sokáig tart és hibákhoz is vezethet a folyamat hossza és monotonitása miatt, ezért egy rendszer, amely előzetesen feldolgozná a felvételeket – kiszűrné és felhívná a figyelmet a gyanúsakra – nagyon hasznos lenne.

A bemutatásra kerülő algoritmus részét képezi a rendszernek, az említett tumorárnyékok, foltok keresésére szolgál a felvételeken. A végleges rendszeren belül több – párhuzamosan futó – algoritmus is foglalkozik egy-egy részprobléma megoldásával, majd ezek kombinációja adja a rendszer végső válaszát.

2. Textúra-alapú folt detektálás

A mammográfia sikere a felvételen látható különféle szövetek elkülöníthetőségén múlik. Alapvető feltételezés, hogy az eltérő szövetek eltérő textúraként jelennek meg a képen. Textúra-analízis segítségével [5] képesek lehetünk részeire bontani a felvételt és osztályozni ezeket a részeket. Ezen részek felhasználhatók diagnózis alkotására, illetve paraméterül szolgálhatnak további algoritmusok számára. Az itt bemutatásra kerülő textúra elemző algoritmus célja egyfajta szövettípus felismerése, a rosszindulatú tumoré.

Ez a foltkeresést végző alrendszer két alapvető részre osztató. Egy durva előszegmentáló részre, valamint egy szegmentálást finomító részre.

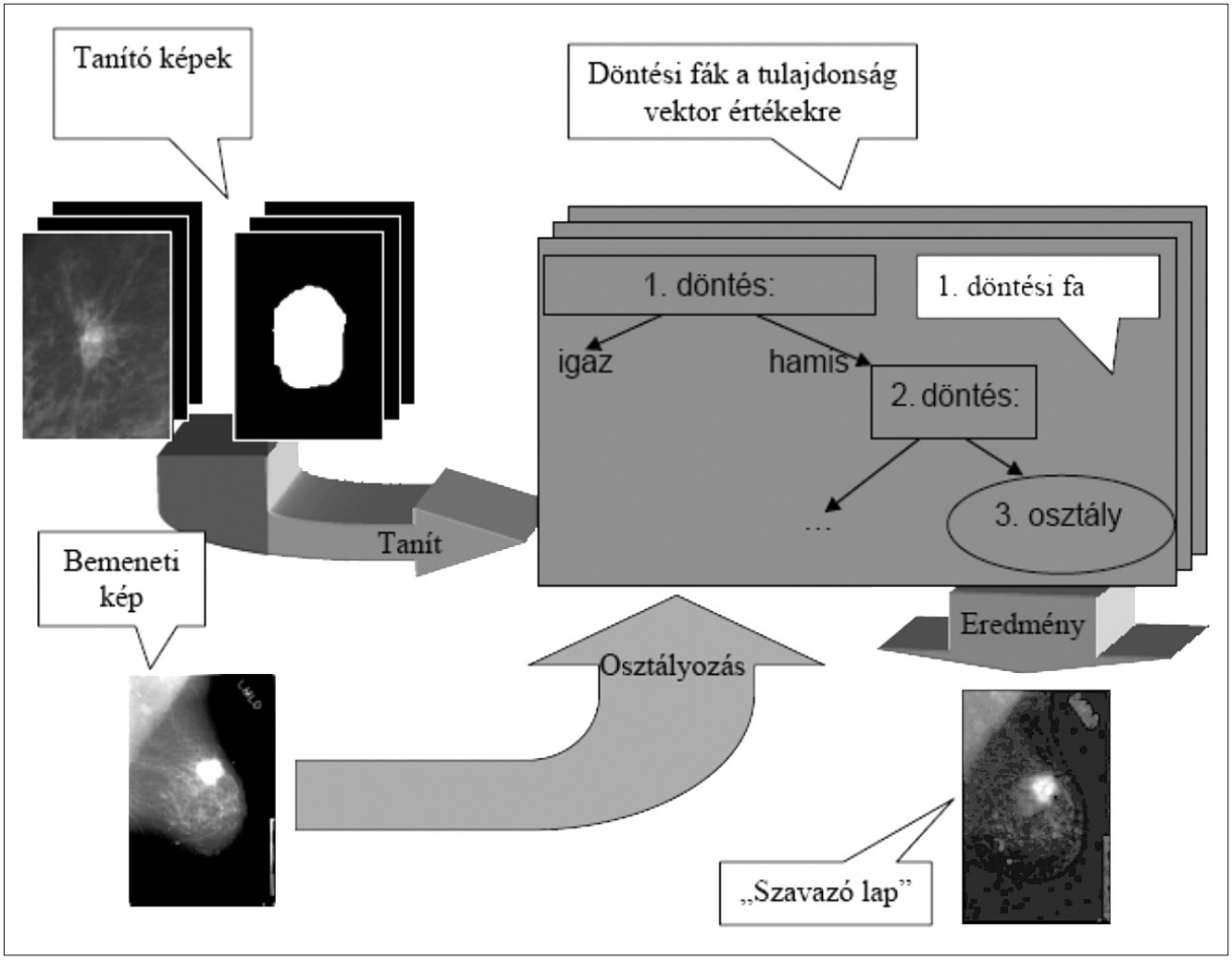
Az algoritmus a felvételt elemi egységekre bontja egy meghatározott ablak méretnek megfelelően. Ebben az ablakban kerülnek kiszámításra a tulajdonság vektor értékek. A jelenlegi implementációban 17 textúra jellemző [5,7,8] paraméterrel írunk le minden textúrát. Ezek a paraméterek ko-okkurencia mátrix [7,8], hisztogram, valamint intenzitás futáshossz jellemzők [5].

Az így kapott tulajdonság vektorokat egy döntési fákból [2,9] álló osztályozó halmaz értékeli ki úgy, hogy egy adott tulajdonság vektort minden egyes döntési fa osztályoz. A döntési fák előzetesen, orvosok által kiértékelt tanító mintákból kerülnek kialakításra CART [2] algoritmussal. A tanítás során az algoritmus különféle foltokat tanul meg megkülönböztetni különféle hátterektől.

Kísérletek során az összes tanító mintából egyetlen központi döntési fa kialakítása olyan bonyolultságú osztályozót eredményezett, mely kezelhetetlenné vált. Ezért a végső megoldásban minden egyes tanító mintából egy különálló fa keletkezik *(1. ábra)*. Ezek a fák a tanítás végeztével, a működés során egy szavazó mechanizmuson keresztül alakítják ki a végleges döntést az adott terület textúra osztályba tartozásáról..

A döntési fák által leadott szavazatok által egy szavazólap jön létre a felvételből. Ez a szavazólap minden egyes pontban tartalmazza az adott pozícióban kiszámított tulajdonság vektorra leadott pozitív szavazatok számát. Minél nagyobb az érték a szavazólapon belül, annál valószínűbb, hogy az adott helyen tumor található.

A szavazólapot ezután egy adaptív küszöböző eljárás – melynek feladata a környezeténél több szavaza-



1. ábra A szegmentálást végző rendszer vázlata

tot kapott részeket megjelölni – bináris maszkká alakítja. Ez a bináris maszk az elsődleges folt jelölteket tartalmazza, de az ablakméretnek megfelelően durva méret és alak információval.

A következő feldolgozási lépés célja ennek a bináris maszknak finomítása. Ezt a lépést végző algoritmus az előszegmentált képet egy Markov-mezőnek [3,4] tekinti, a bemeneti felvételt (ami ebben az esetben a szavazólap) pedig a szegmentált kép és Gauss-zaj keverékének. Az optimális szegmentálást kereső eljárás ICM (Iterated Conditional Modes) [3,4] módszert használ, hogy becsülje a kép MAP (Maximum A Posteriori) szegmentálását.

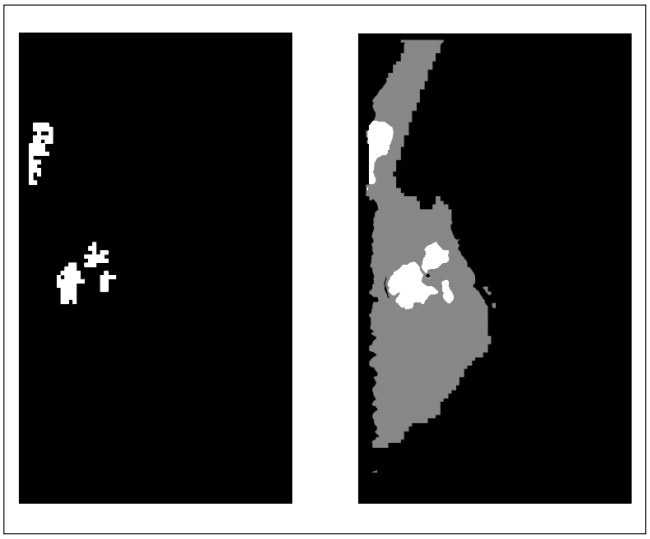
Ez az eljárás egy bináris maszkot ad eredményül mely tartalmazza a végleges folt jelölteket, pontosabb alak és méret jellemzőkkel *(2. ábra)*.

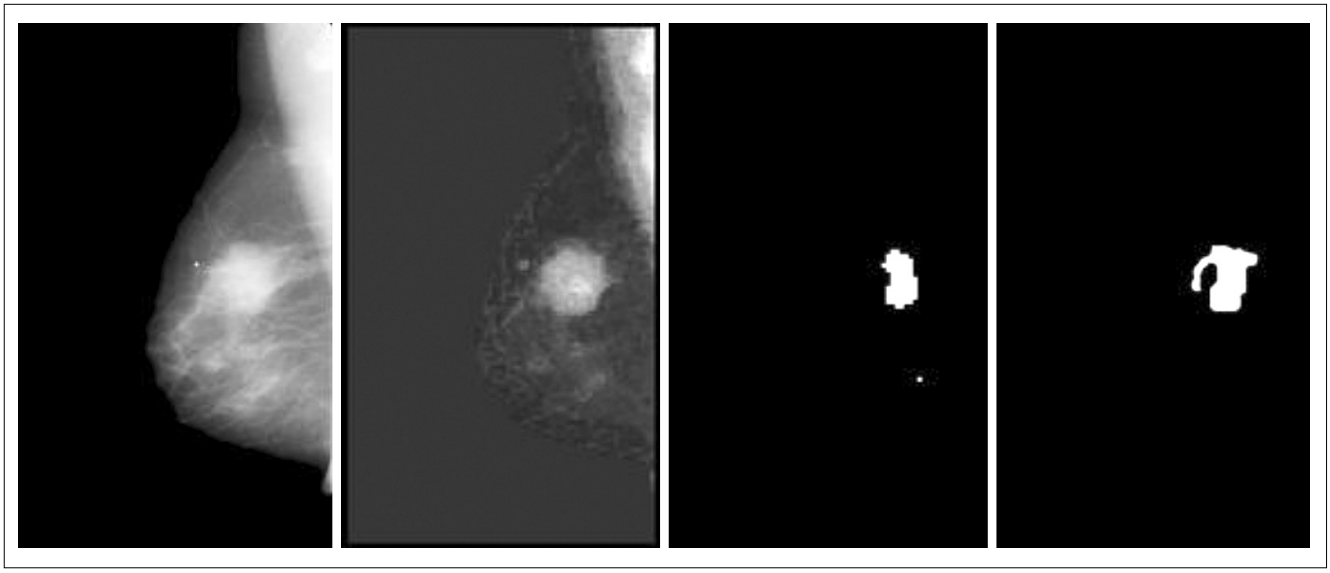
3. Eredmények

Egy olyan komplex rendszer került bemutatásra, melynek célja foltok keresése mammogramokon textúraanalízis segítségével. Az egyes algoritmus lépéseinek bemeneti képei és eredmény képei a *3. ábrán* (a következő oldalon) láthatók.

A rendszer tesztelése – párhuzamosan más foltkereső algoritmusokkal – jelenleg is folyamatosan történik. Az elvégzett nagy méretű tesztek arra vezettek, hogy összességében az ODR rendszer úgy éri el a maximális hatékonyságot a folt keresésében, ha a bemutatott foltkereső algoritmus egy másik foltkereső alrend-

2. ábra A szegmentálás finomítása





3. ábra
Az eredeti felvétel, a szavazólap,
az előszegmentált kép és a finomított eredmény

szer (ami intenzitáskülönbség alapján dolgozik) [6] eredményével kombinálva adja a végeredményt. Ebben az esetben a textúra-elemző rendszer bizonyos speciális esetekben segít, ahol pusztán intenzitáskülönbség alapján nem található meg a folt, ilyenkor segítséget jelent a textúra jellemzők vizsgálata.

A legutóbbi tesztelés során 523 eset került kiértékelésre, ami 2092 felvételt jelent. Az *1. táblázat* (lent) mutatja az eredményeket.

A bemutatott textúra alapú eljárás segítségével sikerült javítani az intenzitás alapú algoritmus teljesítményét. A javulás ugyan csak néhány százalékpontnyi, de ebben a találati tartományban már minden javulás igen nehezen érhető el.

Az elért eredmény különösen értékes abból a szempontból, hogy a fals pozitív jelölések számának csökkenése mellett sikerült elérni. (Az intenzitás alapú algoritmus érzékenységének növelése, amely a rosszindulatú esetek jobb felismerését szolgálhatná, jelentős fals pozitív arány növekedéssel is járna.)

A rosszindulatú esetek felismerésének aránya megfelelőnek mondható (a humán diagnózis sem ér el jobb eredményt), jelenleg a fals pozitív jelzések csökkentése a cél, ez többek közt a két (oldal- és felülnézeti) kép együttes kiértékelésével érhető el.

Irodalom

[1] R. Highnam, M. Brady:
Mammographic Image Analysis,
Kluwer Academic Publishers R, 1999.

[2] Richard O. Duda, Peter E. Hart, David G. Stork:
Pattern Classification, John Wiley & Sons, NY., 2001.

[3] H.D. Li, M. Kallergi, L.P. Clarke, V.K. Jain:
"Markov Random Field for Tumor Detection in Digital Mammography",
IEEE Transactions on Medical Imaging, Vol. 14, No.3, pp.565–576, Sept. 1995.

[4] S.Z. Li:
Markov Random Field Modeling in Computer Vision,
Springer-Verlag, New York [etc.], 1995.

[5] I. Pitas:
Digital Image Processing and Algorithms and Applications, John Wiley & Sons, NY., 2000.

[6] Gábor Takács:
"Computer-Aided Detection of Mammographic Masses",
Proc. of the 12th Mini-Symp., Budapest, Feb. 8-9, 2005.

[7] J. Iivarinen:
"Texture Segmentation and Shape Classification with Histogram Techniques and Self-Organizing Maps",
Acta Polytechnica Scandinavica,
No.95, Helsinki, 1998.

[8] William K. Pratt:
Digital Image Processing,
John Wiley & Sons, NY., 2001

[9] Stuart J. Russel, Peter Norvig:
Artificial Intelligence, Prentice Hall, 1995.

1. táblázat
A foltkereső rendszerek eredményei

	Rosszindulatú eset felismerési arány	Fals pozitív jelölés / kép
Intenzitás alapú algoritmus	93.9%	4.5
Textúra alapú algoritmus	70.0%	3.0
Kombinált algoritmus	95.1%	4.3

Valósidejű háromdimenziós grafika beágyazott környezetben

SZÁNTÓ PÉTER

BME, Villamosmérnöki és Informatikai Kar, Méréstechnika és Információs Rendszerek Tanszék
szanto@mit.bme.hu

Reviewed

Kulcsszavak: 3D megjelenítés, szegmens alapú feldolgozás, FPGA, System-On-Chip

A beágyazott környezetekkel szemben támasztott tipikus követelmények (alacsony ár, kis fogyasztás) következményeként minél hatékonyabb architektúrák kialakítására van szükség, s ez természetesen igaz a megjelenítést végző hardverre is. Jelen cikk egy lehetséges módszert mutat be a háromdimenziós grafikai megjelenítés hatékonyságának növelésére, a szükséges memória sávszélesség csökkentésére.

1. Bevezetés

A felhasználói igények hatására a közeljövőben várhatóan számos újabb helyen merül fel valósidejű és valósághű háromdimenziós grafikai megjelenítés igénye. Ilyen eszközök lehetnek például az egyre nagyobb felbontású és színmélységű megjelenítővel rendelkező mobiltelefonok és digitális személyi asszisztensek (PDA), önálló, TV-vel összeköthető digitális eszközök (set-top-box) vagy akár járművek fedélzeti számítógépei. A 3D megjelenítés hatalmas számításigénye miatt ezekben a viszonylag alacsony – bár rohamosan növekvő – teljesítményű processzorral rendelkező rendszerekben a szoftveres megoldások hamar teljesítőképességük határára érnek; a komplex, valós idejű alkalmazások esetében dedikált, hardveres egységre van szükség.

Azonban nem a CPU teljesítménye jelenti az egyetlen korlátot: a 3D grafika szempontjából a megengedett fogyasztás és az ezzel szorosan összefüggő külső memória sávszélesség szűkös volta talán a legnagyobb megszorítás. Míg asztali számítógépekbe szánt grafikus egységeknél nem ritka a többszáz MHz-es órajelen működő, igen széles memória buszrendszer sem, addig beágyazott rendszerek esetében ennek töredéke áll rendelkezésre a teljes rendszer számára.

Igen fontos tehát egyrészről a rendelkezésre álló erőforrások optimális kihasználása, másrészről pedig egy jól skálázható architektúra kialakítása – hiszen a különböző területre szánt rendszerek teljesítményigénye meglehetősen különböző lehet. További kritériumként merülhet fel a jelenleg elterjedt szoftveres fejlesztői módszerekkel, lehetőségekkel történő kompatíbilitás megtartása, amely az alkalmazások hatékony és gyors adaptációját segíti elő.

2. A 3D megjelenítés alapjai

A 3D megjelenítés alapeleme – akár valós idejű, akár előre renderelt – a háromszög. A virtuális világot leíró modellek saját koordinátarendszerükben, háromszöghálóval közelítve adottak. A felhasznált háromszögek számát a modellezni kívánt objektum bonyolultsága, illetve a megjelenítés finomsága szabja meg: egyszerű esetben (mint például egy kocka) igen kisszámú háromszög felhasználásával tökéletes eredmény érhető el, a bonyolultabb felületek közelítése azonban igen nagyszámú háromszöget is igényelhet.

A megjelenítés [1,2] első lépése – a *transzformáció* – pozíciójuknak és orientációjuknak megfelelően elhelyezi a virtuális világ koordinátarendszerében a lokális koordinátarendszerben definiált objektumokat, majd a kamera helyének és látószögének függvényében a teret a képernyő koordinátarendszerébe transzformálja (a kamera az origóba kerül és a pozitív Z irányba néz). Ugyancsak a csúcspontokon végzendő műveletek közé sorolható a csúcspont alapú *megvilágítás*: ennek során a világban adott fényforrások hatását a háromszögek csúcspontjaira határozzuk meg.

A transzformációt és megvilágítást a *raszterizáció* követi, melynek során a háromszögeket a képernyőt alkotó pixelekre képezzük le, azaz két dimenzióban egyenletesen mintavételezzük.

A *láthatósági vizsgálat* során minden egyes képernyő-pixelre meghatározzuk az azon látható háromszöget, tehát azt, amelyik az adott képernyő pontban a kamerához legközelebb található.

A legutolsó lépés a pixelek színének meghatározása, az *árnyalás* (shading). Ehhez egyrészt felhasználhatók a csúcspont alapú megvilágításnál kiszámított szín értékek: a háromszög belső pontjaiban érvényes szín például a csúcspontokban adott értékek lineáris interpolációjával állítható elő (Gouraud shading). A megjelenítés részletgazdagsága textúrák alkalmazásával tovább növelhető. A legegyszerűbb esetben a textúra az objektum felületi mintájának „fényképe”, amelyet mintegy ráfeszítünk az objektumot meghatározó háromszög vázra. Általánosabban a textúra bármely felületi jellemzőt tároló egy-, két- vagy háromdimenziós tömb, melyet úgy rendelünk a háromszöghöz, hogy annak csúcspontjaiban megadjuk az ott érvényes textúra koordinátákat.

3. Szegmens alapú feldolgozás

Tipikus hardveres megjelenítő egységekben (úgynevezett Immediate Mode Renderer (IMR) megjelenítők) a feldolgozás háromszögről háromszögre haladva történik, a láthatósági vizsgálathoz pedig a Z-buffer algoritmust alkalmazzák. Ennek lényege, hogy minden egyes képernyő pixelhez tartozik egy elem az úgynevezett Z-bufferben (vagy mélységi bufferben), amely az adott időpillanatig feldolgozott, és az adott pixelt fedő háromszögek mélységi (Z) koordinátája közül a minimálist tartalmazza. A Z-buffer – méreténél fogva – a külső memóriában helyezkedik el. Újabb háromszög feldolgozásakor a háromszöget fedő összes pixelre meghatározzuk annak színét, valamint Z értékét (utóbbi is lineárisan interpolálható a csúcspontokban adott Z koordinátákból). A kiszámított mélységi értéket összevetve a Z-bufferben található megfelelő értékkel megállapítható, hogy az új háromszög közelebb van-e a kamerához, mint az eddig feldolgozottak. Amennyiben igen, úgy a Z-buffert és a pixel színeket tároló buffert is frissíteni kell az új adatokkal.

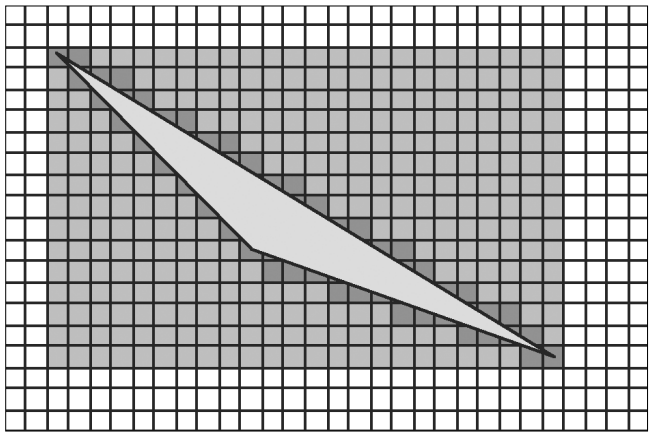
A vázolt algoritmusnak két alapvető problémája van. Egyrészt rendkívül sok memória művelettel jár a pixelenkénti Z-buffer olvasások és esetleges Z-buffer és szín-buffer írások miatt. Másrészt sok felesleges számítást igényelhet, hiszen ha a képet alkotó háromszögek közül a kamerához közelebb levők később érkeznek, akkor sok, már kiszámított pixel színét felülírják. A bemutatott egyszerű algoritmus hatásfoka természetesen nagyban növelhető különböző optimalizációkkal (például Early Z Test, ATI HyperZ [6]), ám ezek továbbra is nagyban függnek a háromszögek feldolgozási sorrendjétől.

Ezzel ellentétben a szegmentált feldolgozás [3] sorrend-független előnyökkel jár. A módszer alapja a kép feldarabolása kis téglalapokra (szegmensek), majd e téglalapok egymástól független feldolgozása. A szegmensek kis mérete lehetővé teszi a Z-buffer áramkörön belül történő megvalósítását, ami egyrészt a külső memóriához fordulások számát jelentősen csökkenti, másrészt igen nagy feldolgozási sebesség elérését teszi lehetővé, hiszen IC-n belül igen széles adatbuszok alakíthatók ki. Nagy teljesítményt igénylő vagy elosztott rendszerek esetén további előny hogy a szegmensek egymástól függetlenül, párhuzamosan feldolgozhatók.

A láthatósági vizsgálat árnyalás előtti elvégzése lehetővé teszi az árnyaló egység hatékony kihasználását, hiszen csak a valóban látható pixelek színét kell meghatározni (a sorrend ilyetén megválasztására IMR esetben is van lehetőség, ami növeli a hatékonyságot, de nem garantálja a 100%-t).

Természetesen e módszernek is vannak hátrányai. A kép első szegmensének feldolgozásakor már rendelkeznünk kell a képet alkotó összes transzformált háromszöggel, hiszen csak így határozható meg biztosan a látható objektum – ez pedig egy képidőnyi késleltetést jelent. Ezen kívül a hatékonyság megőrzéséhez szükség van egy, az IMR megoldásoknál szükségtelen hardver modulra.

Képzeljünk el ugyanis egy kicsi (néhány szegmens nagyságú) háromszöget. Ha ez minden egyes, a képet alkotó több száz szegmensben feldolgozásra kerül, az jelentős mennyiségű felesleges munkát jelent. Célszerű tehát a raszterizáció előtt egy új feldolgozási lépést beiktatni, amely minden egyes szegmensre meghatározza az abban legalább egy pixelt lefedő háromszögeket – így később, a szegmensek feldolgozásakor már csak ezekkel kell foglalkozni. A szegmentálás során két különböző megközelítést alkalmazhatunk: megelégedhetünk a fedett szegmensek körülbelüli meghatározásával, vagy minden háromszögre pontosan meghatározhatjuk a fedett szegmenseket.



1. ábra Szegmentálási módszerek

Előbbi megvalósítható például a háromszög befoglaló téglalapjának felhasználásával; azonban mint az *1. ábrán* látható, aránytalan háromszögek esetén ez a módszer igen rosszul működik (az ábrán például a befoglaló téglalap 370 szegmenst tartalmaz, míg a háromszög csupán 76-ban fed pixeleket).

4. Belső pontok megállapítása

Mielőtt a hardver implementáció részleteivel foglalkoznánk, érdemes a fedés megállapításának kérdéséről szólni, hiszen ez több egység esetében felmerül. A cél tehát egy háromszög belső pontjainak meghatározása.

Definiáljunk ehhez minden egyes háromszög oldalhoz egy, az oldal explicit egyenletéből képzett változót:

$$A(x,y) = (x - x_i) * \Delta y - (y - y_i) * \Delta x \qquad (1)$$

Ez a változó 0 értékű az egyenesen, negatív az egyik és pozitív az egyenes által meghatározott másik félsíkon.

A *2. ábra* a három oldal-változó előjelét mutatja abban az esetben, amikor a csúcspontok *y* koordinátájuk szerint növekvő sorrendbe vannak rendezve, és az (1)-ben szereplő Δ értékeket úgy képezzük, hogy a nagyobb sorszámú csúcspont *x/y* koordinátájából vonjuk ki a kisebb sorszámúét. Ekkor a belső pontokra teljesül az alábbi kifejezés:

$$(s(A_0(x,y)) \; XOR \; s(A_1(x,y))) \; AND \; (s(A_1(x,y)) \; XOR \; s(A_2(x,y))), \quad (2)$$

ahol *s(A(x,y))* az adott oldal *A(x,y)* változójának előjelét jelenti (0 nem-negatív esetben, 1 egyébként). Jól látható az is, hogy a képen *x* irányban lépve az *A(x,y)* változó Δy, míg y irányban lépve Δx mértékben változik.

5. Hardver architektúra

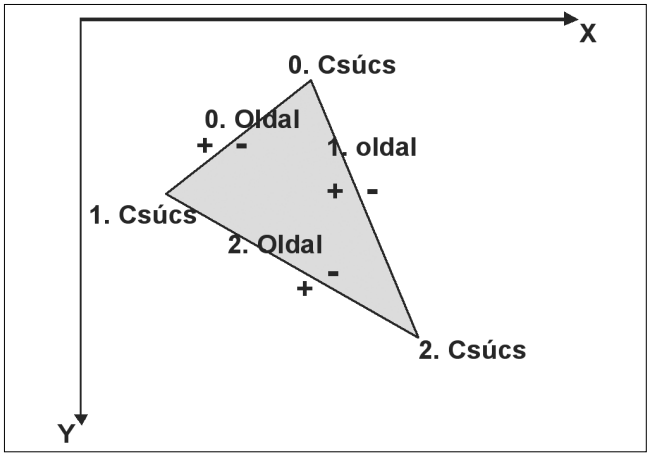
A *3. ábra* egy lehetséges, egyetlen áramkörön belüli System-On-Chip (SOC) rendszer vázlatát mutatja. Ennek része a központi vezérlő szerepét betöltő mikrokontroller, a külső memóriával kapcsolatot tartó memóriavezérlő és a megjelenítésért felelős egység. Természetesen lehetséges a processzor buszra egyéb perifériákat is kapcsolni. A megjelenítő egység ismertetett moduljai a fejlesztés során egy Xilinx XC2V6000 FPGA-ban kerültek megvalósításra, melynek részleteit a hatodik fejezet ismerteti.

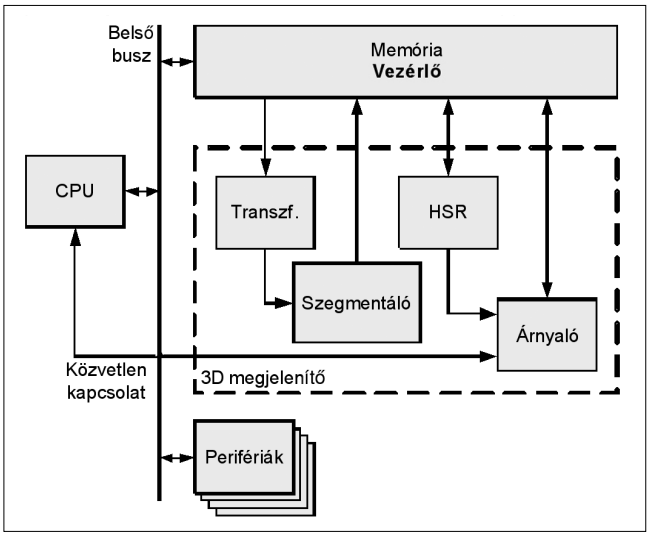
Maga a megjelenítő négy fő részből áll, melyek az egyes megjelenítési lépések elvégzéséért felelősek: a transzformációs *(Transzf)* egység a csúcspontokkal kapcsolatos műveleteket, a *Szegmentáló* a háromszögek szegmensekbe osztását, a *HSR* (Hidden Surface Removal) egység a láthatósági vizsgálatot, míg az *Árnyaló* a pixelek színének kiszámítását végzi.

5.1. Szegmentáló egység

A feldolgozási sorrendben ez az egység közvetlenül a transzformáció után következik. Bemenetei a transzformált csúcspontok *x* és *y* koordinátái, míg kimenete minden szegmenshez egy lista, amely az adott szegmensben levő háromszögek sorszámát tárolja. A kimeneti lista nagysága a megjeleníteni kívánt kép bonyolultságától függ, viszonylag nagy mérete miatt külső memóriában kapott helyet, így fontos kritérium hogy a feldolgozási láncban következő, láthatósági vizsgálatot végző HSR egység részéről ez a lista minél egyszerűbben feldolgozható legyen.

2. ábra Belső pontok megállapítása





3. ábra SOC rendszer

A hatékonyság maximalizálása érdekében a szegmens mérete képkockák között változtatható, ennek kezelésére mind a szegmentáló, mind pedig a HSR egység képes. A minimális szegmens méret 32x16 pixel, mind vízszintes, mind pedig függőleges irányban ezen méret többszörösei választhatók, egymástól függetlenül.

A szegmentáló három részből áll. A bemeneti fokozat végzi a bemeneti *x, y* koordinátákból a szegmens generátor számára szükséges adatok kiszámítását (Δ értékek, *A(x,y)* változók kezdeti értéke). A szegmens generátor egység a háromszög által fedett szegmenseken lépdel végig, s órajelenként generál egy érvényes szegmens koordináta kimenetet. A kimeneti fokozat a szegmens generátor kimeneti adatait felhasználva memóriacímeket és vezérlőjeleket állít elő a kimeneti lista felépítéséhez.

5.1.1. Bemeneti Fokozat

A bementi egység két nagyobb feldolgozó láncra bontható. A transzformációs egységgel történő kiegyensúlyozás megvalósítására a bemeneti adatok (háromszög csúcspont *x, y* koordináták) egy FIFO-ba kerülnek.

Az első feldolgozó lánc ezekből a csúcspontokból generál érvényes háromszögeket: ehhez 1, 2 vagy 3 új csúcspontra van szükség (például diszkrét háromszögek esetén új háromszöget három új csúcspont határoz meg, míg háromszög-szalag esetén elegendő egyetlen új csúcspont). Az érvényes háromszögek csúcspontjait *y* koordinátájuk szerint sorba rendezi.

A második feldolgozó lánc egyrészt az (1)-ben szereplő Δ értékeket határozza meg, másrészt kiszámítja a három oldalegyenes *A(x,y)* változóit a kezdő szegmens (amelyikben a háromszög 0. csúcspontja van) két felső csúcspontjában. A szegmens határok a pixel középpontok között találhatók. A pipeline 9 fokozata a következő funkciókat valósítja meg:

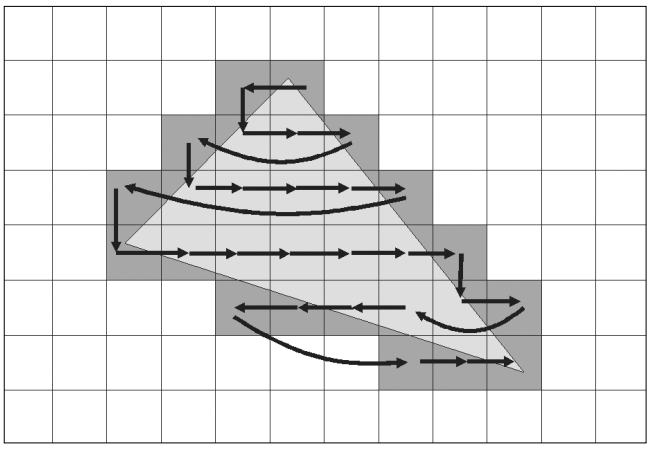
– bemenet multiplexálása;
– csúcspont szegmensének meghatározása (2 fokozat);

– háromszög-csúcspontok távolságának meghatározása a szegmens bal felső csúcsától, (1)-ben szereplő Δ értékek kiszámítása;
– (1)-ben szereplő részszorzatok meghatározása (2 fokozat);
– Δ értékek szorzása szegmens mérettel, *A(x,y)* változók meghatározása a bal felső szegmens csúcspontban;
– *A(x,y)* változók kiszámítása a jobb felső szegmens csúcspontban.

A hardver erőforrások csökkentésének érdekében a pipeline egyszerre egy oldal adatait képes feldolgozni (ezért került multiplexer az első pipeline fokozatba), tehát egy háromszög összes adatának előállítása három órajelet vesz igénybe. A szorzás műveletek az órajel frekvencia maximalizálása érdekében két ütem alatt történnek.

5.1.2. Szegmens generátor

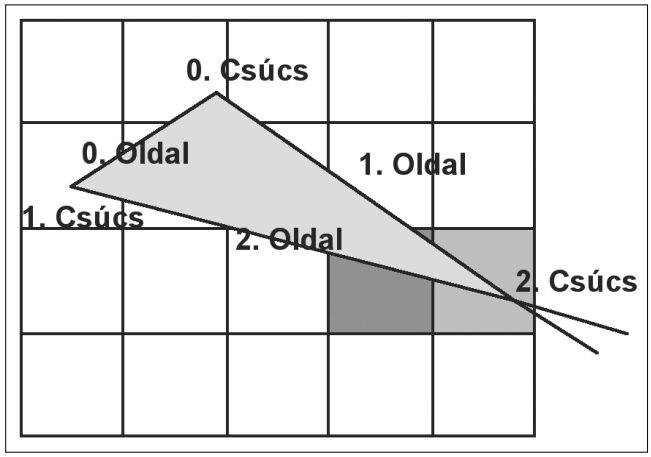
A fedett szegmensek koordinátáit előállító egység végiglépdel a feldolgozás alatt levő háromszög által fedett szegmenseken. Első megfontolásra durva felbontású raszterizáló egységnek hihetnénk, de mint látható lesz, ez sajnos nem így van. A feldolgozás a 0. háromszög-csúcspont szegmensében kezdődik, és mint minden szegmens-sorban, a jobbra lépéssel indul: az egység addig halad jobbra, amíg szükséges. A jobbra lépés befejeztével két eset lehetséges. Amennyiben a kezdő szegmenstől balra is található fedett szegmens, az algoritmus a kezdő szegmenstől eggyel balra található szegmensre ugrik, és a szükséges ideig lépdel balra. Ha jobbra lépés után nem kell ugrani, vagy befejeződött a balra lépés is, a következő szegmens-sor feldolgozása következik. Új szegmens-sorba lépésnél az algoritmus garantáltan a háromszög által fedett szegmensbe lép. Az elmondottakat szemlélteti a *4. ábra*.



4. ábra Szegmentálás

A jobbra lépések triviális esete amikor a szegmens jobb oldali csúcspontjainak egyike a háromszög belső pontja (pl. (6,4) szegmens). A további esetek figyelembevételéhez metszéspont értékeket generálunk minden szegmens-oldal és minden háromszög-oldal kombinációjával. Ezeknek előállítására ugyancsak az (1)-ben definiált *A(x,y)* értékek használhatók, hiszen egy háromszög akkor metsz egy szegmens-oldalt, ha a szegmens-oldalt meghatározó szegmens-csúcspontokban különbözik az adott oldal *A(x,y)* értékének előjele.

Az *5. ábra* a hasonló esetek egyikét mutatja. A sötétszürkével jelölt szegmensből jobbra kell lépni, annak ellenére, hogy a megfelelő szegmens-csúcspontok nem belső pontjai a háromszögnek. Ugyanakkor az 1. és 2. háromszög-oldalaknak van metszéspontja a jobb oldali szegmens-oldallal, tehát ennek alapján a lépési döntés meghozható. Ugyanezek a metszéspontok azonban a világosszürke szegmens esetében is megtalálhatók, itt mégsem szabad jobbra lépni; mégpedig azért mert az algoritmus elérte azt a szegmenst, amely tartalmazza a megfelelő (jelen esetben 2.) háromszög-csúcspontot.



5. ábra Jobbra lépés

Mint már említettük, a jobbra lépés befejezése után balra lépések következnek, amennyiben szükséges. A szükségesség eldöntéséhez a kezdő szegmens szolgál információval: amennyiben ott balra lépés is szükséges, akkor van szükség a jobbra lépés utáni visszaugrásra. A visszaugrás a kezdő szegmenstől balra található szegmensre történik, s innen folytatódik az esetleges balra lépési sorozat.

A jobbra és balra lépések végeztével a következő szegmens-sor feldolgozása kezdődik meg. Annak érdekében, hogy az algoritmus sor-lépésnél mindig biztosan háromszög által fedett szegmensbe kerüljön, a vízszintes irányú lépdelés során folyamatosan vizsgálja, hogy az adott szegmens megfelelő sor-lépési pont-e. Pozitív eredmény esetén a szegmens alsó csúcspontjaiban érvényes *A(x,y)* értékek elmentésre kerülnek, így ezek a későbbi sor-lépéshez felhasználhatók. Annak eldöntésére, hogy egy szegmens megfelelő-e sor-lépéshez, ugyancsak az *A(x,y)* értékekre van szükség: amennyiben az alsó szegmens-csúcspontok belső pontok, vagy az alsó szegmens-oldalnak metszéspontja van a megfelelő háromszög-oldalakkal, akkor a szegmens jó lépési pont. A háromszög feldolgozása akkor ér véget, amikor a legalsó háromszög-csúcspont szegmens-sorában járunk, és sem jobbra, sem pedig balra nem kell már lépni.

5.1.3. Kimeneti fokozat

A kimeneti fokozat minden egyes szegmenshez egy, a külső memóriában tárolt, 32 szavas tömbökből álló láncolt listát állít elő (szegmens lista). Egy 32 szavas blokk első 31 eleme háromszög sorszámokat tárol, míg az utolsó elem a következő 32 szavas tömb memória címét mutatja. A szegmensekhez tartozó első tömb fix címen található, míg a további tömböket dinamikusan foglaljuk, a szegmensben található háromszögek számának függvényében. A lista ilyetén kialakítása lehetővé teszi a HSR egység részéről a nagyobb egységekben történő beolvasást, de mégsem bánik túlságosan pazarlóan a memóriával.

A lista előállításához szükség van egy belső memóriára is, amely minden szegmenshez tárolja a következő (külső memória) írási címet. Új kép feldolgozásának megkezdésekor e címeknek a szegmensekhez tartozó első tömb első elemére kell mutatniuk; erről két, speciális háttér-háromszög gondoskodik, melyek feldolgozása alatt az összes cím alaphelyzetbe állítható. Ezen kívül csak a következő szabad 32 szavas blokk memória címét szükséges tárolni, amely új kép esetén a képen található szegmensek száma szorozva a tömb mérettel (32), és minden egyes új blokk foglalásakor inkrementálódik.

A külső memória foglaltsága esetén természetesen nem lehetséges az adatok kiírása, így ebben az esetben a kimeneti egység a memória felszabadulásáig leállíthatja a szegmens generátor működését.

5.2. HSR egység

A HSR egység a láthatósági és stencil tesztek elvégzéséért felelős, szegmensről szegmensre haladva dolgozza fel a képet. Az egyes szegmensekben azok a háromszögek kerülnek feldolgozásra, amelyek megtalálhatók a szegmenshez tartozó szegmens listában. A HSR egység minden háromszög esetén a szegmens összes pixelét megvizsgálja, függetlenül attól, hogy az a háromszög által fedett-e, vagy sem. Ez természetesen rontja a feldolgozás hatékonyságát, ugyanakkor determinisztikus működési időt eredményez, ami mind a cella, mind pedig a bemeneti fokozat vezérlését egyszerűsíti, valamint lehetővé teszi utóbbi egység erőforrás-igényének csökkentését.

A HSR egység egy bemeneti egységből és több, ugyanolyan felépítésű cellából áll. Alapesetben (minimális szegmens méret) a szegmens egy-egy sora van az egyes cellákhoz rendelve. A vízszintes irányú szegmens méret növekedésével ez nem változik, míg függőleges méret növekedése esetén N cella esetén minden N-edik szegmens sort az N-edik cella dolgoz fel.

5.2.1. Bemeneti egység

A bemeneti egység több műveletvégzőből áll, melyek alkalmasak a szükséges bemeneti adatok előállítására. Ezek egyrészt az Szegmentáló esetén már ismertetett, a háromszög oldalaihoz tartozó oldalegyütthatók és Δ értékek, másrészt a mélységi (Z) értékek számításához szükséges változók. A belső pontokra a Z értékeket a csúcspontokban adott értékekből lineáris interpolációval határozzuk meg, felhasználva a sík egyenletét:

$$z(x,y) = E_z * x + F_z * y + G_z = \frac{A_z}{C_z} * x + \frac{B_z}{C_z} * y + \frac{D_z}{C_z}, \qquad (3)$$

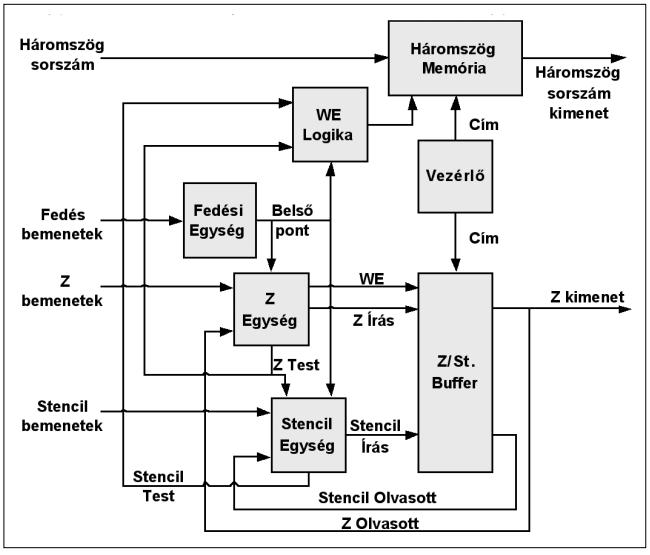
ahol a megfelelő együtthatók a csúcspont értékekből származtathatók:

$$\begin{aligned} A_z &= (z_1 - z_2) * (y_1 - y_0) - (y_1 - y_2) * (z_1 - z_0) \\ B_z &= (x_1 - x_2) * (z_1 - z_0) - (z_1 - z_2) * (x_1 - x_0) \\ C_z &= (x_1 - x_2) * (y_1 - y_0) - (y_1 - y_2) * (x_1 - x_0), \\ D_z &= A_z * x_0 + B_z * y_0 + C_z * z_0 \end{aligned} \qquad (4)$$

Mivel a cellák a bemeneti egységhez képest kétszeres órajellel járnak, és a minimális szegmens szélesség (azaz egy cella minimális feldolgozási ideje) 32 órajel, így a bemeneti egységnek 16 órajel áll rendelkezésre a szükséges adatok előállításához.

5.2.2. HSR cella

A tényleges feldolgozást a cellák végzik: a hozzájuk rendelt pixeleken végighaladva egyrészt megvizsgálják, hogy az belső pont-e, kiszámítják a megfelelő Z értéket (ez *x* és *y* irányokban haladva mindössze egy-egy összeadást igényel), és elvégzik a Z, valamint stencil tesztet. Egy cella vázlatát az *6. ábra* mutatja.



6. ábra HSR Egység

A feldolgozó modulok mellett minden cellához két, független memória is tartozik. Az egyikben a pixelenkénti Z és stencil értékeket tároljuk (Z/St Buffer), míg a másik kimeneti memóriaként funkcionál: minden pixelhez az azon látható háromszög sorszámát tárolja (Háromszög Memória).

Z egység

Az egységnek két különböző működési módja van az átlátszatlan, és az áttetsző háromszögek esetére. A már említett Z-buffer memóriában minden pixelhez két

érték tartozik (egymást követő páros és páratlan címeken), melyekből átlátszatlan esetben csak a páros címek kerülnek felhasználásra, míg áttetsző háromszögek feldolgozásakor a teljes memóriára szükség van.

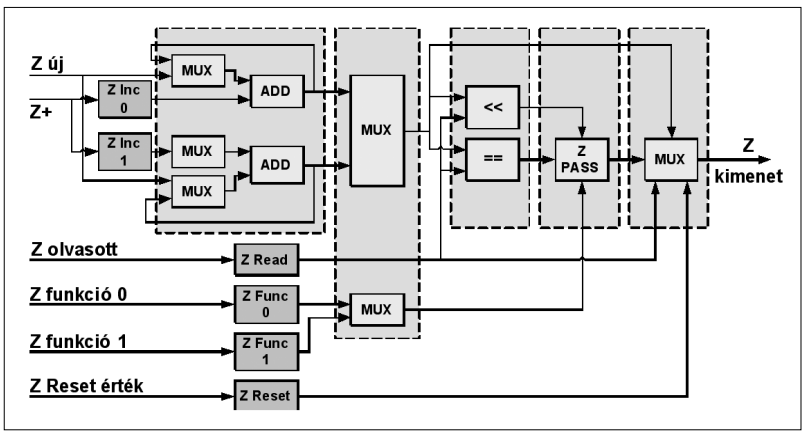
Átlátszatlan esetben a már ismertetett Z-buffer algoritmus kerül megvalósításra, azaz a feldolgozás alatt levő háromszög Z értékeit a Z-buffer tartalmával kell összehasonlítani. A szegmensek feldolgozása az átlátszatlan háromszögekkel kezdődik.

Az áttetsző háromszögek feldolgozása ([4]) ugyanakkor – lévén az áttetszőség sorrend-függő művelet – sorba rendezést igényel. A Z egység képes ezen háromszögek több menetben történő rendezésére és feldolgozására, pixelhelyes átlátszóságot érve el (a tipikus megjelenítők nem rendelkeznek hasonló funkcióval, ott a körülbelüli – nem pixelhelyes – sorba rendezés a CPU-ra hárul, ami a tipikus SOC rendszerek igencsak véges számítási kapacitását figyelembe véve nem feltétlenül tehető meg). Minden egyes feldolgozási menetben meghatározzuk azt a háromszöget, amely a kamerától a legmesszebb, de a már kiszínezett háromszögnél közelebb helyezkedik el. Ehhez – a legegyszerűbb megvalósítást tekintve – minden menetben fel kell dolgozni az összes áttetsző háromszöget.

Konkrét esetet (első menet) vizsgálva, a Z-buffer páros címein az adott pixelen látható átlátszatlan háromszög Z értéke található. Az áttetsző háromszögeket feldolgozva tehát keressük az ennél kisebb Z értékűeket, ami egy komparálást jelent pixelenként. Megszokott funkciójára felhasználva a Z-buffer páratlan című helyeit egy további komparálással lehetőség nyílik az áttetsző háromszögek közül a legtávolabbi kiválasztására. A Z teszt eredménye tehát akkor lesz igaz értékű, ha mindkét komparálás eredménye igaz. A következő menetben a Z-buffer egy pixelhez tartozó értékei funkciót cserélnek, hiszen ekkor a páratlan helyeken találjuk az első (már feldolgozott) áttetsző háromszög Z értékét, míg a páros címek felhasználhatók az újabb maximumkeresésre.

A Z egység vázlatos felépítését az *7. ábra* mutatja. A megfelelő működési frekvencia eléréséhez – csakúgy, mint a cella többi egysége – a Z Egység is pipeline szervezésű. Az áttekinthetőség érdekében az ábra csak az egyes fokozatok funkcionalitását mutatja (szaggatott téglalapok), a bennük levő regisztereket nem.

7. ábra Z egység



Egy háromszög feldolgozásának kezdetekor a háromszöghöz tartozó kezdeti Z értéket (*Z új*), az interpolációhoz szükséges értéket (*Z+*) és a vezérlő jeleket (Z funkció, reset érték) töltjük be. Az interpolálásra két, párhuzamosan működő összeadó szolgál (első fokozat), melyek két órajelenként, de egymáshoz képest egy órajellel eltolva írják saját regiszterüket.

Átlátszatlan esetben a két egység két, egymást követő pixel Z értékeit tartalmazza, míg áttetsző esetben ugyanazt a Z értéket. A következő fokozat multiplexere órajelenként váltakozva választ az előző fokozat két kimenete közül, majd a komparálási fokozatban megtörténik a Z-bufferből kiolvasott megfelelő értékkel történő összehasonlítás. Az összehasonlítás eredménye a beállított komparálási függvénynek (mindig igaz, soha nem igaz, kisebb, kisebb-egyenlő, nagyobb, nagyobb-egyenlő) megfelelően megadja hogy a Z teszt igaz-e. Ennek, valamint a fedési és stencil teszt eredményének felhasználásával már eldönthető, hogy milyen értéket kell a Z-bufferbe visszaírni:

– a feldolgozás alatt levő háromszög új értékét, amennyiben mindhárom teszt igaz;
– a reset értéket,
  – ha a Z teszt hamis, de belső pontról van szó és resetelni kell a Z-buffert,
  – nem belső pontról van szó és resetelni kell a Z-buffert;
– a kiolvasott értéket minden egyéb esetben.

A Z-buffer resetelésére (ami tipikusan a lehető legnagyobb értékkel való feltöltést jelenti, de ez nem megkötés) minden egyes új kép feldolgozásának kezdetekor szükség van.

Átlátszatlan esetben tehát a Z egység órajelenként képes egy pixelt feldolgozni, míg áttetsző esetben két órajel szükséges egy pixel feldolgozásához.

Stencil egység

A stencil funkció pixelek maszkolását teszi lehetővé, aminek megvalósítására egy pixelenkénti érték szolgál. Amennyiben a stencil teszt eredménye hamis egy adott pixelen, úgy annak színe nem módosul. Eme funkció segítségével számos különböző effektus megvalósítására nyílik lehetőség. Kompozíció során például több, különböző kameraállásból készített 2D vagy 3D képet illeszthetünk össze; a Stencil-buffer felhasználásával az újabb részleteket egy jól definiált, maszkolt területre helyezhetjük be. Így egyszerűen megvalósítható például szövegek megjelenítése, de ez a funkció megfelelő, például visszapillantó tükrök megjelenítésére is.

Az utóbbi esetben a vezető nézete az egyik 3D kép, míg a visszapillantó tükör pozíciójából készített kép a Stencil-buffer által kijelölt részre kerül. További, bonyolultabb effektusok (például árnyékok ([5]), tükröződések, objektum körvonalak) is meg-

valósíthatók a Stencil-buffer segítségével, ám ezeknek részletesebb ismertetése túlmutat jelen cikk keretein.

A stencil teszt során használt értékek:

- Stencil-buffer értéke (StBuff))
- Stencil referencia érték (StRef)
- Olvasási maszk (RMask)
- Írási maszk (WrMask)
- Komparálási funkció (COMP)
- Művelet (OP)

A fenti, zárójelben megadott jelölésekkel élve a Stencil-teszt eredménye a következő (& bitenkénti ÉS műveletet jelöl):

$$(StRef \ \& \ RMask) \ COMP \ (StBuff \ \& \ RMask) \qquad (5)$$

A komparálási funkció a Z egységnél már felsoroltak valamelyike lehet.

A Stencil-bufferbe írandó értéket az alábbi összefüggés adja meg:

$$(StBuff \ \& \sim WrMask) \mid (WrMask \ \& \ OP(StBuff)), \qquad (6)$$

ahol & bitenkénti ÉS, | bitenkénti VAGY, ~ pedig bitenkénti negálás műveletet jelent. Ellentétben a Z-bufferrel, a Stencil-buffer új értékkel történő írása nem csak abban az esetben lehetséges, ha a teszt eredménye igaz. Lehetőség van ugyanis külön művelet beállítására az alábbi esetekre:

- Stencil teszt hamis,
- Stencil teszt igaz, Z teszt hamis,
- Stencil teszt igaz, Z teszt igaz.

A lehetséges, a Stencil-bufferből olvasott értéken végrehajtható műveletek (OP):

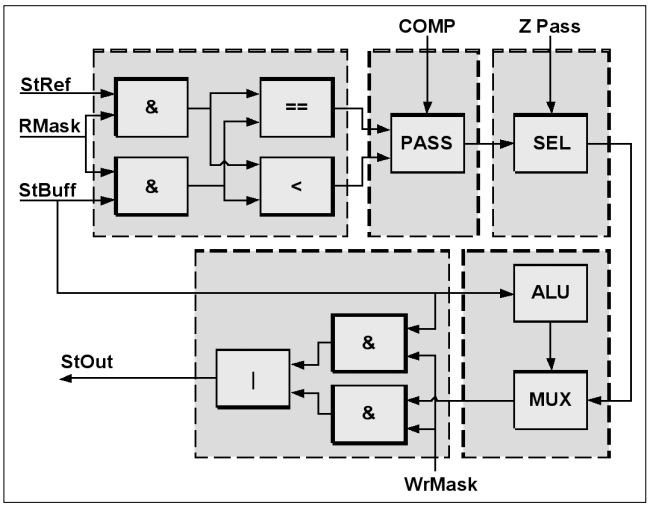
- nulla beírása,
- olvasott érték megtartása,
- bitenkénti invertálás,
- inkrementálás,
- inkrementálás szaturációval,
- dekrementálás,
- dekrementálás szaturációval.

A szintén pipeline felépítésű stencil egység blokkvázlatát a *8. ábra* mutatja, a szaggatott téglalappal körbekerített részek egy-egy fokozatot jelentenek.

Működése megfelel az eddig leírtaknak, talán csak annyi megjegyzés szükséges, hogy a választható műveletek eredményeit az ALU egység párhuzamosan generálja, s ezekből egy multiplexer választja ki a megfelelőt. A multiplexer vezérlőjelét a megelőző pipeline fokozat állítja elő, a stencil, Z és fedési teszteknek megfelelően.

6. Eredmények

A hardver modulok realizálása Xilinx XC2V6000-4 típusú, 6 millió kapu bonyolultságú FPGA (Field Programmable Gate Array) eszközön történt. A rendelkezésre álló, FPGA-ban implementálható processzor (Xilinx MicroBlaze) 80-100 MHz körüli órajelet képes elérni ezen eszközben, így a tervezett moduloknál is ez az órajel tartomány volt a cél.



8. ábra Stencil Egység

Az Indexelő egység összes modulja képes elérni a 100 MHz-es órajelet, így maximális számítási kapacitása 100 millió szegmens másodpercenként. A másodpercenként feldolgozható háromszögek száma a háromszögek által fedett szegmensek számától függ, a csúcsteljesítmény 33 millió háromszög másodpercenként (ekkor minden háromszög három vagy kevesebb szegmensben fed pixelt, és a szegmentáló bemeneti fokozata a limitáló tényező), míg az elméleti minimum 390 ezer háromszög másodpercenként (32x16 pixeles szegmens, 640x480 pixel felbontású kép és fél képernyőt fedő háromszögek).

A HSR Egység bemeneti egysége 100 MHz-es órajelen működik, míg maguk a cellák 200 MHz elérésére képesek. A jelenlegi, 8 cellát tartalmazó implementációban ez 1600 millió átlátszatlan pixel, és 800 millió áttetsző pixel feldolgozását jelenti másodpercenként. Mivel a szegmenseken belül a feldolgozás ideje a fedett pixelek számától független, az effektív kitöltési sebességet (a feldolgozott háromszög-pixelek számát) a szegmensekben átlagosan fedett pixelek aránya szabja meg, ami megfelelő szegmens méret megválasztásával maximalizálható.

Irodalom

[1] A. Watt: 3D Computer Graphics, Addison-Wesley, 2000.
[2] Szirmay-Kalos László: Számítógépes grafika, ComputerBooks, 2001.
[3] H. Holten-Lund: Design for scalability in 3D computer graphics architectures, Ph.D. Thesis. Technical University of Denmark, 2001.
[4] P. Diefenbach: Pipeline Rendering: Interaction and Realism Through Hardware-Based Multi-Pass Rendering, Ph.D. Thesis. University of Pennsylvania, 1996.
[5] Kilgard, M. J., Everitt C.: Optimized Stencil Shadow Volumes. Game Developer Conference, 2003.
[6] ATI Technologies, Performance Optimization Techniques for ATI Graphics Hardware with DirectX 9.0 ATI Radeon SDK, http://www.ati.com/developer/

Adaptív dokumentumelemzés információkinyeréshez

DEZSÉNYI CSABA, MÉSZÁROS TAMÁS, DOBROWIECKI TADEUSZ

BME, Villamosmérnöki és Informatikai Kar, Méréstechnika és Információs Rendszerek Tanszék
dezsenyi@mit.bme.hu

Reviewed

Kulcsszavak: összetett dokumentumelemzés, elemzési terv készítése, információkinyerés

Manapság egyre nagyobb szerepet kapnak a különböző információ- és tudásmenedzsment alkalmazások, mind az ipari, mind a tudományos területeken. Komoly kihívást jelent viszont a folyamatosan növekvő méretű információtengerben megtalálni a számunkra érdekes információszeletet, kinyerni a számunkra értékes tudást. Ennek megfelelően az alkalmazásokban különösen fontos elemek lettek a különböző információkinyerési módszerek és technikák, melyek segítségével természetes nyelvű dokumentumokból tudunk releváns információt kiemelni.

1. Bevezetés

Ahhoz, hogy megfelelő teljesítményt érjünk el, nem elég egy-egy izolált algoritmust felhasználni, több különböző dokumentumelemzési módszert kell összehangoltan alkalmazni. Egy gazdasági tényeket kinyerő alkalmazásban például a következő feldolgozási lépésekre lehet szükség:

(1) szöveges tartalom kivágása a HTML oldalból,
(2) szavak és mondatok szegmentálása,
(3) személy- és intézménynevek felismerése,
(4) szófajtani elemzés és végül
(5) mondattani elemzés.

Az ilyen és ehhez hasonló, összetett információkinyerést alkalmazó rendszerek fejlesztése eddig elszigetelten történt, az elemzések megvalósítását pedig az egyedi alkalmazásoknak megfelelően alakították ki, különböző architektúrákra, adatmodellekre és implementációkra alapozva.

A bemutatásra kerülő dokumentumelemző keretrendszer célja az, hogy egységes elméleti és szoftver keretet nyújtson olyan alkalmazások fejlesztéséhez, melyben természetes nyelvű szövegek összetett elemzésére és feldolgozására van szükség. A keretrendszerben tetszőleges dokumentumelemzési feladat megvalósítható, így az ilyen alkalmazások alapvető platformjaként képes szolgálni.

A keretrendszer alapötlete az, hogy egy összetett dokumentumelemzési feladatot automatikusan dekomponálunk kisebb és egyszerűbb műveletekre, majd így elvégezve az elemzést, az eredményt konzisztensen egyesítjük. A tervezés fázisai között az egyik legnagyobb kihívás annak az adatmodellnek a kialakítása, mely segítségével a független elemzőmodulok eredményei konzisztensek és összefüggőek lesznek a végrehajtás után. Az elméleti alapokról és a megalkotott adatmodellről bővebben [1]-ben olvashatunk.

Jelen cikkben a keretrendszer rövid bemutatása után egy új adaptív dokumentumelemzési megoldást ismertetünk.

2. A dokumentumelemző keretrendszer bemutatása

Az információkinyerés teljes folyamata három fázisra osztható: dokumentumbeszerzés, dokumentumelemzés, és végül az információkinyerés. Első lépésként be kell szerezni azokat a dokumentumokat a forráskörnyezetből, melyek az alkalmazás által igényelt információt tartalmazhatják. Ez legegyszerűbb esetben egy lokális fájlrendszerből való iterált beolvasást jelent. Bonyolultabb megoldást kíván, ha a megfelelő dokumentumokat az interneten kell megkeresni és onnan letölteni. Ilyen intelligens dokumentum kereső és beszerző rendszer fejlesztése szintén a kutatás része, melyről bővebben [2]-ben olvashatunk. Az első fázis eredménye a beszerzett dokumentum, valamilyen kiindulási struktúrába öntve.

Ezután többféle módon elemezni kell a dokumentumot, aminek eredményeképpen az eredeti forrás különböző strukturált reprezentációi állnak elő, melyeket *nézeteknek* nevezünk. Ezeknek a nézeteknek kell tartalmaznia az alkalmazás által igényelt információelemeket. Az elemzés során létrejött nézetek a bemenetei a harmadik fázisnak, ahol az alkalmazás számára érdekes információt le lehet kérdezni belőlük.

A rendszerbe illeszthető dokumentumelemző modulok az alapvető építőkövek, segítségükkel lehet összetett elemzési sémákat kialakítani és így bonyolult dokumentumelemzési feladatokat elvégezni. Számos eszköz vizsgálatára alapozva kialakítottunk egy absztrakt dokumentumelemző meta-modellt. A keretrendszer ennek segítségével egységesen tudja kezelni az egyes modulokat, implementációtól függetlenül. Az interfészek és adatmodellek kialakításánál különös figyelmet fordítottunk arra, hogy tetszőleges dokumentumelemzési művelet megvalósítható legyen.

Egy dokumentumelemző modul feladata az, hogy a bemenetként kapott dokumentumban felismerjen bizonyos elemeket, majd ezeket egy kimeneti dokumentumba transzformálja. Mind a bemeneti, mind a kimene-

ti dokumentumok speciális formátumúak az elemzők számára, ezek a már említett nézetek. Egy létrejövő nézetet tekinthetünk úgy, mint az eredeti forrás egy bizonyos típusú információs vetületét, mely valamilyen ismert struktúrában tartalmazza az elemző által azonosított információ-elemeket. Minden nézetnek van egy típusa, mely megmondja, hogy milyen fajta információt tartalmaz, illetve amely definiálja annak struktúráját (sémadefiníciók segítségével). A nézetek megvalósítása tipikusan XML-el lehetséges, azaz egy nézet, egy XML dokumentum.

Mivel egy elemzőmodul nézeteket állít elő, a bemenetei pedig szintén nézetek, az egyes modulok fel tudják használni mások eredményeit. A keretrendszer bemenete egy kezdeti nézet, amely az elemezni kívánt forrásdokumentumot tartalmazza valamilyen (alkalmazás függő) kezdeti struktúrába öntve. A teljes elemzési folyamat eredménye az egyes modulok által előállított nézetek szemantikailag összefüggő halmaza, melyet *nézethálózatnak* hívunk [1].

3. Dokumentumelemzési séma tervezése

A beszerző rendszer szolgáltatja az elemző keretrendszer számára a forrásdokumentumokat (mint kiindulási nézetek), míg az információkinyerés fázisa előírja, hogy milyen nézetek szükségesek ahhoz, hogy az alkalmazás számára igényelt információt ki tudjuk nyerni. A keretrendszer feladata tehát, hogy előállítsa a megfelelő nézeteket a különböző elemzőmodulok alkalmazásával. Ehhez ki kell választani a szükséges modulokat, meg kell tervezni a futási szekvenciát, végre kell hajtani az elemzési folyamatot és létre kell hozni az elemzés eredményét, azaz a teljes nézethálózatot.

Problémát okoz azonban az, hogy a megfelelő modulok kiválasztása, illetve a végrehajtási sorrend nem triviális, mert pl. modulok igényelhetik mások kimenetét, esetleg több fajta bemeneten képesek dolgozni, vagy egy tipikus eset lehet, hogy egy fajta nézetet többféleképpen tudunk előállítani. Az általunk kidolgozott módszer alapja, hogy klasszikus MI tervkészítő algoritmusokat alkalmazunk (például STRIPS alapú részben rendezett tervkészítő algoritmust [3]) az elemzési sémák részben vagy teljesen automatizált készítéséhez.

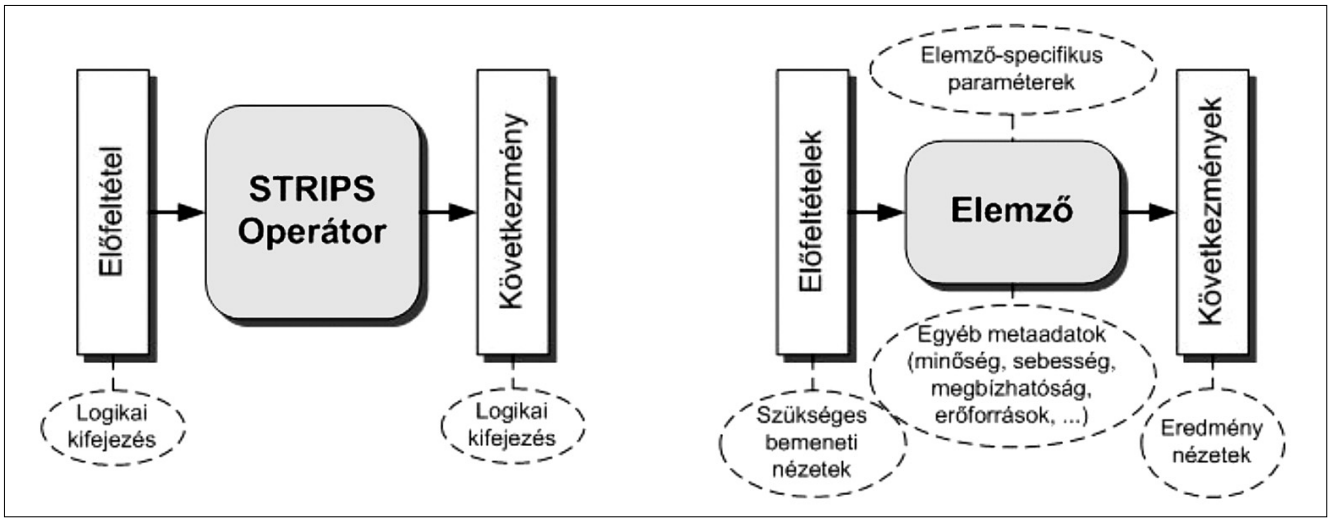
Mivel a klasszikus MI tervkészítési problémák és a keretrendszerben lévő elemzési séma problémaköre nagymértékben analóg, kézenfekvő megoldásnak tűnik a már kidolgozott módszerek és algoritmusok adoptálása. A keretrendszer egyes elemei, elnevezései könnyen megfeleltethetők az MI tervkészítés terminológiájában használatos fogalmaknak.

A tervben lévő operátorok (melyek tk. cselekvések, állapotváltozást okoznak) itt az elemzőmodulok lesznek, melyek meglévő nézetekből új nézeteket állítanak elő *(1. ábra)*. Egy modul előfeltétele a futáshoz igényelt bemeneti nézeteire vonatkozó logikai feltételekből áll, míg a hatás az eredményképpen létrejövő nézetek logikai leírása.

A terv kiindulási állapota az, amikor egy új dokumentum kerül a rendszerbe, mint kiindulási nézet, a célállapot pedig akkor valósul meg, mikor az összes olyan nézet előállt, ami szükséges az információkinyeréshez. A tervkészítő algoritmus által létrehozott részben rendezett terv itt a végleges tervként szolgál, mivel az egyes modulok párhuzamosan is tudnak futni, nincs szükség linearizálásra. A fogalmak megfeleltetése után már könnyű alkalmazni a megfelelő MI tervkészítő algoritmust, a működési mechanizmus ugyan az lesz, mint általános esetben *(2. ábra)*.

A terv kiindulási állapota két absztrakt elemzőt tartalmaz: „start” és „cél”. A start lépés a kiindulási nézeteket állítja elő és nincs előfeltétele, a cél pedig az alkalmazás által igényelt nézeteket definiálja előfeltételek formájában, nyílván következmény része nincsen. Az algoritmus feladata az, hogy a start és a cél közötti utat megtalálja megfelelő elemzőmodulok illesztésével. Regresszív tervkészítő esetében az illesztés a céltól visszafelé történik, azaz első lépésként a célhoz keres

1. ábra STRIPS operátor és Elemző



olyan alkalmas elemzőmodulokat, melyeknek a következmény része kielégíti a cél előfeltételeit. Amennyiben a tervnek létezik megoldása, akkor ezt folytatva előbb-utóbb eljutunk a startig.

Hagyományos tervkészítő algoritmusok esetében fontos kérdés az, hogy az egyes operátorok ne rontsák el mások előfeltételeit (például egyik előállít egy szükséges feltételt, viszont később egy másik mellékhatásként törli azt).

Ezt az úgynevezett védett szakaszok elméletével építik bele az algoritmusokba. Ennek lényege, hogy az operátorok sorrendezését kényszerítik úgy, hogy ne fordulhasson elő ilyen szituáció. Mivel a keretrendszerben lévő elemzők előfeltételei és következményei kizárólag ponált elemeket tartalmazhatnak (inkrementális jelleg: mindig nézetet állít elő egy elemző, sose töröl), ezért a rendszer kedvező tulajdonsága, hogy ilyen jellegű probléma nem fordulhat elő.

4. Nyitott kérdések és az implementáció jellemzői

Általános esetekben az adaptáció tökéletesen működik, de összetettebb konfigurációk esetében adódhatnak olyan konfliktus szituációk, melyek automatikus feloldása nehézségeket okozhat. Például egy tipikus eset, amikor egy fajta nézetet két féle elemző is elő tud állítani. A tervkészítő algoritmus szempontjából ekvivalens a két elemző használata, ám a valós kimenetel nagyon különbözhet, egyes esetekben az egyik, máskor a másik lehet a jobb hatásfokú.

Vajon ilyenkor az volna célszerű, ha a tervkészítéskor eldöntenénk egy prioritás jellegű választással? Vagy nyitva hagyva a választást, a végrehajtás egy bizonyos fázisában lenne érdemes dönteni, hogy melyik fusson? Esetleg mindkettő futtatása után próbálja meg a rendszer a jobb eredményt kiválasztani, vagy egy optimális uniót képezni belőlük? A kutatás jelenlegi fázisában kézi konfliktusfeloldást használunk, így ezek a fontos és érdekes kérdések jelenleg még nyitottak, a teljesen automatikus és optimális tervkészítés megoldása még fejlesztés alatt áll.

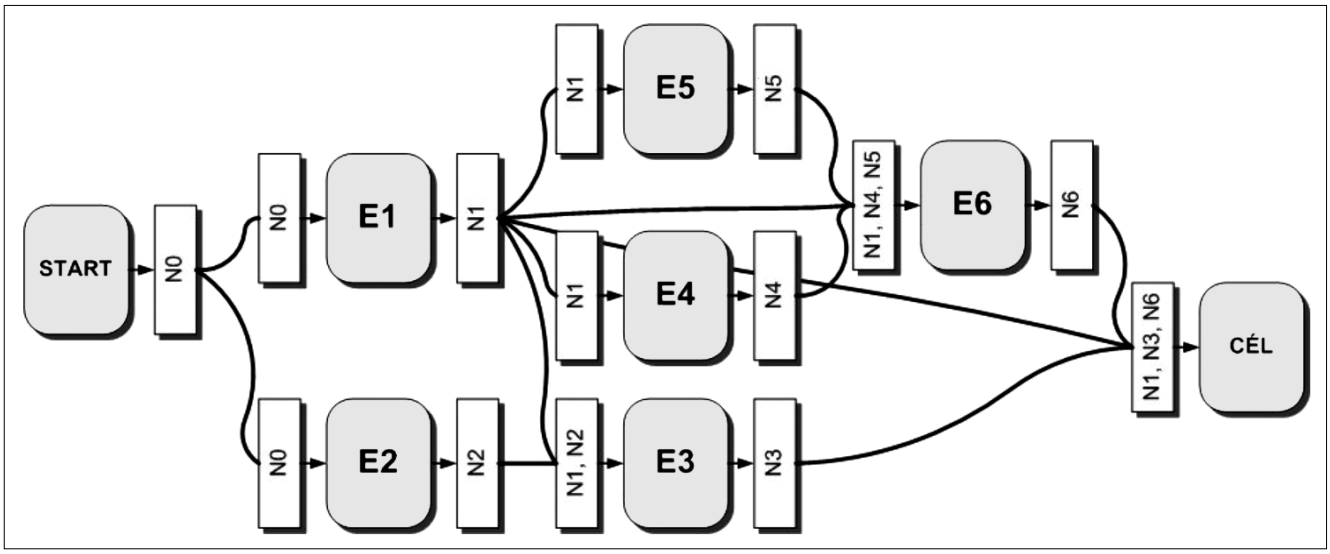
A dokumentumelemző keretrendszer prototípus implementációja elkészült és néhány egyszerű elemzőmodul is készült tesztelés céljából. A rendszerben használatos adatmodellek és metaadat kezelés teljes egészében XML alapú.

Mivel az alapötlet, az elméleti keret és maga az implementáció sikeressége is egyaránt a gyakorlati használhatóságon múlik, ezért a rendszer tesztelése és értékelése folyamatosan történik. Emellett a kezdeti tapasztalatokra alapozva mindenképpen ígéretesnek mutatkozik a kutatás és számos valós célalkalmazás fejlesztését is tervbe vettük.

Irodalom

[1] Cs. Dezsényi, T. Mészáros, T. P. Dobrowiecki: "Parser Framework for Information Extraction", Proc. of EUROFUSE Workshop on Data and Knowledge Engineering, September 22-25, Warszawa, Poland, 2004.
[2] Cs. Dezsényi, P. Varga, T. Mészáros, Gy. Strausz, T. P. Dobrowiecki: „Ontológia Alapú Tudástárház Rendszerek", Proceedings of Networkshop 2003 Conference, Pécs, April 14-17, 2003.
[3] S. Russell, P. Norvig: Artificial Intelligence. A Modern Approach. Prentice Hall Inc., 1997.

2. ábra Elemzési terv példa: egyszerű ténykinyerés
(E1: tokenizáló, E2: nyelvfelismerő, E3: indexelő, E4: morfológiai elemző, E5: névfelismerő, E6: mondatelemző, N0: kiindulási nézet, N1: szavak és mondatok, N2: dokumentum nyelv, N3: index, N4: szavak morfológiája, N5: felismert nevek, N6: elemzett mondatok)



Közösségi döntések implementációjának új megközelítése

KOVÁCS DÁNIEL LÁSZLÓ

BME, Villamosmérnöki és Informatikai Kar, Méréstechnika és Információs Rendszerek Tanszék
dkovacs@mit.bme.hu

Reviewed

Kulcsszavak: intelligens ágensek, korlátos optimalizálás, implementációs elmélet

Az intelligens rendszerek jelentős szerepet játszanak mindennapjainkban. Ennek ellenére mindmáig nincsen olyan átfogó rendszerspecifikációs elv, amely lehetővé tenné e rendszerek egységes tervezését, és elemzését. Az intelligens rendszerek tervezése, és elemzése tehát mind a mai napig esetleges, megoldandó feladathoz igazított, többnyire ad-hoc módon történik.

1. Bevezetés

A fent említett problémára keres megoldást a játék, ágens, és evolúciós elméletek egyesítése [1]. Az egyesítés kulcsa, pontosabban az intelligens rendszerek jóságának általános mércéje a racionalitás egy újfajta definícióján, a korlátos optimalitáson alapszik. Egy intelligens rendszert (ágenst) akkor tekintünk korlátosan optimálisnak, ha a környezetében kivitelezett cselekvéseit egy olyan program szerint választja meg, amelynél nincs jobb azok között, amelyeket futtatni képes [2]. Ahhoz tehát, hogy módunkban álljon ilyen rendszerekről érdemben beszélni, szükségünk lesz az ágensek programjának egy használható, absztrakt modelljére. Ebből a célból kerül bevezetésre a virtuális haszon fogalma, mint az ágens-programok modelljének egy központi összetevője.

Az intelligens rendszerek döntési mechanizmusának, avagy az ágensek programjának ily módon történő modellezése lehetővé teszi az ágensekből alkotott közösségek működésének (pontosabban a közösségi döntések implementációjának [3]) egy újfajta, az eddigieknél hatékonyabb megközelítését.

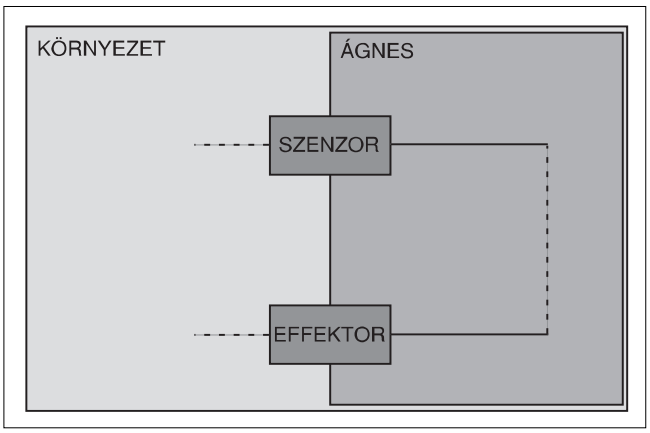
2. Hogyan modellezzük az intelligens rendszereket?

Tegyük fel, hogy az intelligens rendszerek modellezhetők ágensként *(1. ábra)*. Egy *ágens „bármi lehet, amit úgy tekintünk, mint ami szenzorai segítségével érzékeli környezetét, és effektorai segítségével megváltoztatja azt”* [4]. Ennek a föltételezésnek az adja a létjogosultságát, hogy – túl azon, hogy intuitív, és kellőképp általános – lehetővé teszi az intelligens rendszerek környezetének, szenzorainak, és effektorainak konkrét megfeleltetését. Azaz tetszőleges intelligens rendszer esetén megadható a környezet, az érzékelő szervek, és a beavatkozó szervek konkrét megfeleltetése.

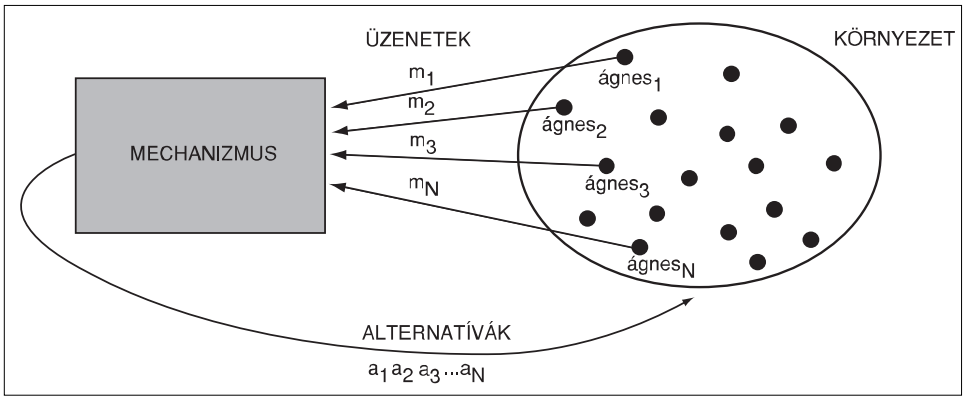
A környezetébe ágyazott ágens minden pillanatban a következő cselekvés kiválasztásának problémájával szembesül (ahol magát a tétlenséget is egyfajta cselekvésnek tekinthetjük). Nem-triviális környezetek esetén *tervkészítésre* van szükség e döntések hatékony meghozásához. Több-ágenses környezetben az egyes ágensnek ráadásul még a többi ágens viselkedését is figyelembe kell vennie ahhoz, hogy hatékony cselekvési tervet készíthessen. E helyzetek modellezésére nyújt alkalmas keretet a játékelmélet [5], amely az ágensek egymásra gyakorolt hatását játékosok közt fellépő stratégiai kölcsönhatásoknak tekinti egy *játékban*, ahol az ágensek a *játékosok*, terveik pedig a *játékosok stratégiái* [6]. Mindazonáltal a játékelmélet csak az egyed szempontjából, nem pedig közösségi szinten vizsgálja a döntéshozás kérdését.

Jelenleg az implementációs elmélet (mint a játékelmélet egyik legújabb ága) foglalkozik közösségi döntési helyzetek modellezésével. Az ágenseket együttesen *közösségnek* tekinti, melynek céljai egy *közösségi döntési szabály* (KDSZ) formájában összegezhetők, azaz egy olyan leképzés formájában, amely a releváns rejtett paraméterek alapján előállítja a végkimeneteleket. *Magyarán az ágens-közösséget úgy tekinti, mint ami – kollektív entitásként – egy adott KDSZ-nek megfelelően cselekszik.* A KDSZ tehát úgynevezett közösségi alternatívákat (például végkimeneteleket) állít elő a közösségen belüli ágensek privát információja (például egyéni – végkimenetelek felett értelmezett – preferen-

1. ábra Intelligens ágensek általános felépítése



ciái) alapján. Az egy-értékű KDSZ-t szokás *közösségi döntési függvénynek* (KDF) is nevezni. Az implementáció problémája ekkor a következőképp foglalható össze: *Adható-e olyan mechanizmus, amely az ágensek adott elv szerint hozott döntései mellett a közösségi optimumot implementálja?* [3]



2. ábra Az implementáció problémája

A *2. ábra* valamivel részletesebben szemlélteti az implementáció problémáját: adott tehát egy Tervező, akinek a feladata az, hogy – az ágensek adott viselkedését feltételezve – olyan mechanizmust hozzon létre, amely implementál egy adott KDSZ-t. Pontosan fogalmazva: a cél egy olyan mechanizmus létrehozása, amely adott környezet mellett ugyanazokat az $a_1, a_2, a_3, ..., a_N$ alternatívákat (például végkimeneteleket) eredményezi, mint egy adott KDSZ, feltéve, hogy az *1, 2, 3, ..., N* ágensek egy adott *S* játékelméleti megoldási elvnek (például domináns stratégiák, Nash-egyensúly [7]) megfelelően választják $m_1, m_2, m_3, ..., m_N$ üzeneteiket (avagy a mechanizmusban játszott stratégiáikat). Amennyiben az adott feltételek mellett létezik ilyen mechanizmus, úgy a KDSZ-t *S-implementálható*nak nevezzük.

A fenti megközelítésnek több előnye is van. Képes például szociális intézmények, külsődleges társadalmi ráhatások, ágensek közötti megállapodások modellezésére. Számos közgazdasági, politikai helyzet modellezésére alkalmas. Ismeretes például, hogy, ha az ágensek által követett *S* játékelméleti megoldási elv a domináns stratégiák (azaz, ha az ágensek mindig a domináns stratégiájukat választják, amely minden más stratégiájuknál jobb eredményt ad függetlenül attól, hogy a többi ágens milyen stratégiát választ), akkor kizárólag diktatórikus KDF-ek implementálhatók. A diktatórikus KDF mindig egy adott ágens – kimenetelek felett értelmezett – preferenciáinak kedvez, azaz olyan kimenetelt eredményez, ami az adott ágens hasznát maximálja. Ennek az igen „negatív” eredménynek az egyik legfőbb oka az, hogy nem minden játékban van a játékosoknak domináns stratégiája.

Az előnyök mellett természetesen a megközelítésnek több hátránya is van. Nem közgazdasági, vagy társadalmi helyzetekben, hanem például az informatikában, mesterséges intelligens rendszerek (szoftver ágensek, robotok stb.) tervezésekor a Tervezőnek *közvetlen* ráhatása van a rendszer belső felépítésére, működésére, programjára). Az *S* megoldási elv viszont csak egy *közvetett* feltételezés erre vonatkozólag. Nyilván ennek az okai az implementációs elmélet társadalomtudományi gyökereiben keresendők, ahol az ágensek (vállalatok, emberek stb.) nem megváltoztatható módon adottak. Felvetődhet tehát a kérdés, hogy miért is kellene az ágenseknek éppen egy adott *S* elvnek megfelelően működnie? Az ilyen, és ehhez hasonló kérdésekre az implementációs elmélet sajnos már nem ad magyarázatot. Hátránynak tekinthető továbbá, hogy az ágensek egy *központi* mechanizmuson keresztül kénytelenek cselekedni, ami ráadásul *globális hozzáférésel* bír az ágensek környezetéhez. Ez általában véve egy irreális feltevés, főként intelligens ágens-rendszerek tervezésekor, ahol az ágensek működése legtöbbször *decentralizált*, és a környezethez (például Internet, Mars felszíne) való hozzáférés többnyire csak lokális.

További hátrány, hogy ahhoz, hogy egy KDSZ implementálható legyen, általában igen sok *speciális feltételnek* kell eleget tennie (monotonitás, ordinalitás, indíték kompatíbilitás stb.), ami igencsak leszűkíti az implementálható KDSZ-ek körét. Végül, de nem utolsó sorban hátrány, hogy általános esetben csakis *approximatív implementáció* lehetséges, azaz tetszőleges – a speciális feltételeknek eleget tevő – KDSZ implementálható, de csak megközelítőleg, valamekkora hibával. Ezt nevezik *virtuális implementációnak* [8].

3. Közösségi döntések implementációjának új megközelítése

Az implementációs elmélet fentebb felsorolt hátrányainak kiküszöbölését tűzi ki célul a virtuális haszon alapú döntéshozás elve. A környezet legyen egy *játék*, melyben az ágensek legyenek a *játékosok*, terveik pedig a *játékosok stratégiái*, továbbá minden játékoshoz tartozzon egy *haszonfüggvény* is, amely minden lehetséges stratégia-kombináció esetén megadja az adott játékos környezetben vett valós hasznosságát. Ez lesz tehát az a hasznosság, amit az adott játékos valójában elér, ha mindenki a stratégia-kombinációban neki megfelelő stratégiát játssza. Minden egyes játékos rendelkezzen továbbá *a játék egy belső reprezentációjával,* azaz a játék egy modelljével (beleértve a többi ágenst, stratégiáikat, és haszonfüggvényeiket).

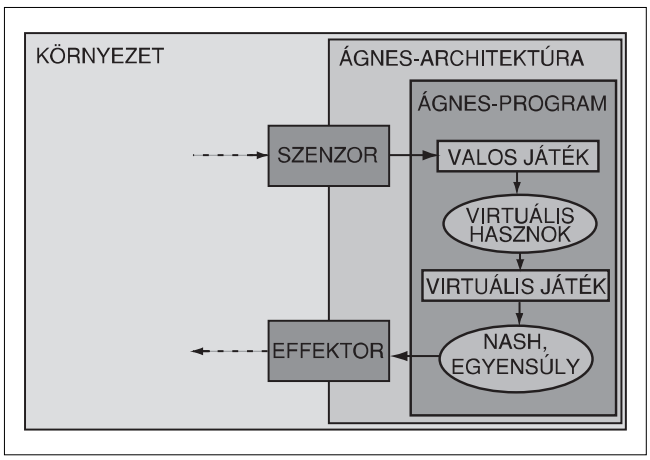
Az ágensek felépítése ekkor legyen a következő: legyen adott egy *architektúrájuk*, és egy *programjuk*, ahol az architektúra legyen felelős a program futtatásáért, a program pedig a játék belső reprezentációja alapján válassza ki az ágens által játszott stratégiát. Az ágensek programjának modellje ekkor legyen a követ-

kező: minden játékosnak legyen egy *virtuális haszonfüggvénye,* amely a játék belső reprezentációjában lehetséges összes stratégia-kombinációhoz egy-egy virtuális haszonértéket rendel. Ekkor a játékos programja felfogható úgy, mint ami éppen azt a stratégiát választja ki a játékos számára, amit a virtuális játék (melyben a játékosok stratégia-kombinációkhoz tartozó haszna az adott játékos által részükre feltételezett virtuális haszonfüggvényük által adott) valamely Nash-egyensúlya ír elő (az a stratégia-kombináció, amelytől egyik játékosnak se éri meg egyoldalúan eltérni). Ezt a döntési mechanizmust szemlélteti *3. ábra:*

Az előbbiek alapján jól látható, hogy az ágensek működését (programját) sikerült explicit módon modellezni. Ebből következően a Tervező immár decentralizált, lokális környezeti hozzáférésű mechanizmus formájában implementálhat egy-egy KDSZ-t azáltal, hogy megadja az ágensek ehhez szükséges architektúráját, és programját (azaz lényegében a virtuális haszonfüggvényeiket). Bizonyítást nyert, hogy teljes információs játékoknak megfelelő környezetekben tetszőleges KDF, megkötés nélkül, egzakt módon implementálható bináris virtuális haszonfüggvények felhasználásával [9]. A bizonyítás konstruktív, így tehát adott ágens architektúrák mellett megadja azokat a virtuális haszonfüggvényeket, melyek mellett a fentebb leírt ágens-működés közösségi szinten éppen egy tetszőleges választott KDF-et implementál.

Az új megközelítés hátrányának tekinthető, hogy viszonylag magas absztrakciós szinten modellezi az ágensek működését, s így még további kutatás szükségeltetik ahhoz, hogy egy-egy konkrétan adott ágens-architektúrának (pl. JADE) is megfeleltethető legyen. Előnye viszont, hogy közvetlenül, egzakt módon, megkötés nélkül tervezhetünk általa ágens-közösségeket. Ennek következtében bizonyítható módon válik implementálhatóvá az „optimális" közösségi működés (például Pareto optimális – azaz ha nincs a közösségnek olyan része, amely jobban jár akkor, ha eltér stratégiájától, miközben a többiek egyike se jár rosszabbul –, vagy korlátosan optimális). Érdekes, és fontos ágens-társadalmi jelenségek is modellezhetővé válnak továbbá.

3. ábra Ágensek működésének újfajta megközelítése



Például az ágensek közti kooperáció (ahol a kooperáló ágenseknek azonos a virtuális haszonfüggvénye), vagy éppen az áldozathozatal jelensége (ahol az „áldozatkész" ágens virtuális haszna ott magas, ahol valós haszna alacsony) stb. Ezen felül a játékelméletben már jól ismert típus-központú megközelítés [10] felhasználásával (ahol a játékosok típusa most az ágens architektúrájának, és programjának együtteseként értendő) kezelhetővé válik a nem teljes információs játékoknak megfelelő környezetek esete is.

4. Összefoglalás

A cikkben bemutatott virtuális haszon alapú döntéshozási elv lehetővé tette, hogy létrehozzuk az ágensek, és az ágensközösségek működésének egy olyan absztrakt, magas-szintű modelljét, amely segítségével az eddigieknél sokkalta hatékonyabban válik megoldhatóvá a közösségi döntések implementációjának problémája. A további kutatás az említett modell már meglévő, alacsonyabb szintű ágens-modellekkel való összekapcsolását; a nem teljes információs játékoknak megfelelő probléma-környezetek vizsgálatát; és az elképzelés – intelligens rendszerek egységes tervezésére, és elemzésére irányuló – átfogó rendszerspecifikációs elvbe történő integrációját tűzi ki célul.

Irodalom

[1] D. L. Kovács:
"Intelligens rendszerek egységes tervezése,"
Híradástechnika, 2004/10. pp.29–38.
[2] S. Russell, D. Subramanian:
"Provably bounded-optimal agents,"
Journal of AI Research, Nr.2,1995, pp.1–36.
[3] R. Serrano: "The Theory of Implementation of Social Choice Rules," SIAM Review, 46:377-414, 2004.
[4] S. Russell, P. Norvig: Artificial Intelligence:
A Modern Approach, Prentice Hall, 1995.
[5] J. von Neumann, O. Morgenstern:
Theory of games and economic behavior,
Princeton University Press, 1947.
[6] M. Bowling, R. Jensen, M. Veloso:
"A Formalization of Equilibria for Multiagent planning,"
in Proc. of IJCAI'03 Worskshop, August 2003.
[7] J. F. Nash: "Non-cooperative games,"
Annals of Mathematics, 54(2), 1951. pp.286–295.
[8] D. Abreu, A. Sen:
"Virtual Implementation in Nash Equilibrium,"
Econometrica, Nr.59, 1991. pp.997–1021.
[9] D. L. Kovács:
A general model to strategy selection in games,
Technical report, BUTE-DMIS, Hungary, May 2004.
[10] J. C. Harsányi: "Games with incomplete information played by Bayesian players I-II-III,"
Management Science,14. pp.159–182; 320–334;
486–502, 1967–1968.

Hol járunk?
Helymeghatározás autonóm járművekben

TÓDOR BALÁZS

BME, Villamosmérnöki és Informatikai Kar, Méréstechnika és Információs Rendszerek Tanszék
balazs.todor@mit.bme.hu

Kulcsszavak: robotnavigáció, lokalizáció, Particle Swarm Optimization

Az elmúlt években egyre gyakrabban találkozhatunk önállóan mozgó járművekkel, gépekkel, és néhány év múlva már akár a saját házunkban is használhatunk ilyeneket. Ha közelebbről megvizsgáljuk az autonóm robotokat, az esetek döntő többségében a megoldandó probléma legmélyén a navigációt találjuk. Alaposabb vizsgálódást végezve láthatjuk, hogy ezek a feladatok két, nagy csoportra bonthatóak: egyesek olyan megoldásokat igényelnek, amelyeknek csak az ütközésmentes mozgást kell biztosítani a környezetben, míg a bonyolultabbak már célirányos navigációt követelnek meg – természetesen az akadályok kikerülése ekkor is megmarad alapvető követelménynek.

1. Az autonóm jármű

Hogyan is néz ki egy autonóm jármű belülről? Először is szüksége van néhány érzékelőre, amelyekkel az akadályokat észreveheti, illetve amelyekkel a mozgás célpontjait is felismerheti. Mivel a navigáció mindkét kategóriájában alapvető feladat az ütközésmentes mozgás, ezért legtöbbször távolságmérőket (például ultrahangos, lézeres, infravörös) szokás a robotokra szerelni.

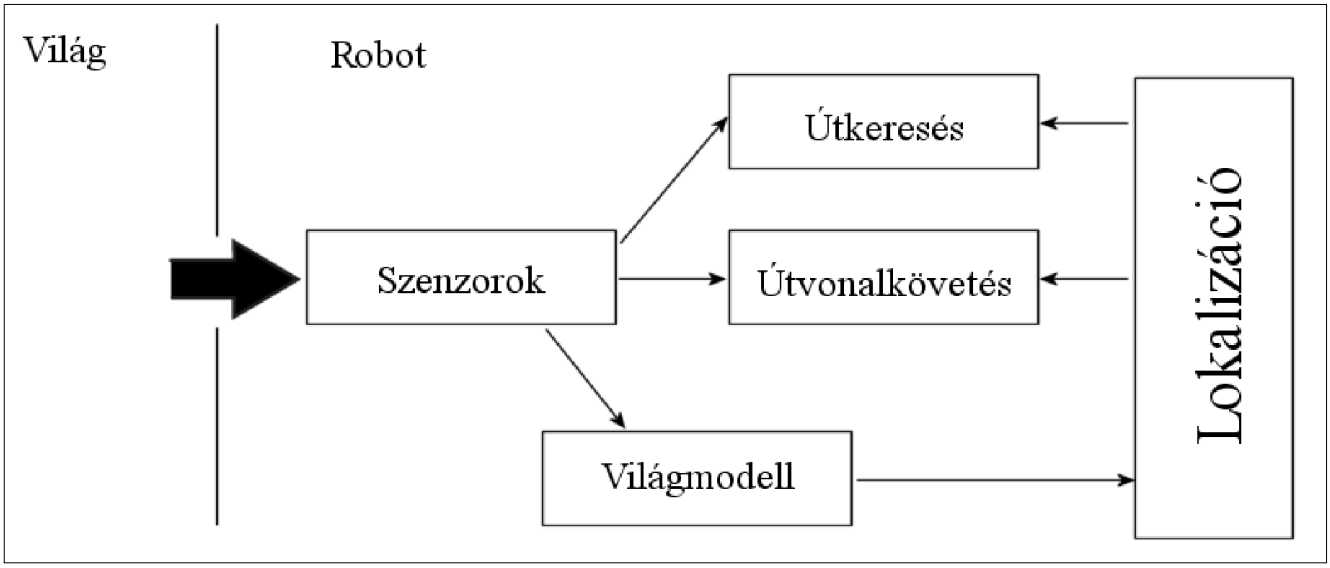
A szenzoraink jeleiből a járművünknek fel kell tudnia építeni a környezet valamilyen modelljét, ami csak a lényeges információkat tartalmazza, azokat viszont – a navigáció szempontjából – a lehető legegyszerűbb formában. Ezt nevezzük világmodellnek, és az útkeresés, illetve az ezt követő útvonalkövetés is ezen a modellen fog dolgozni.

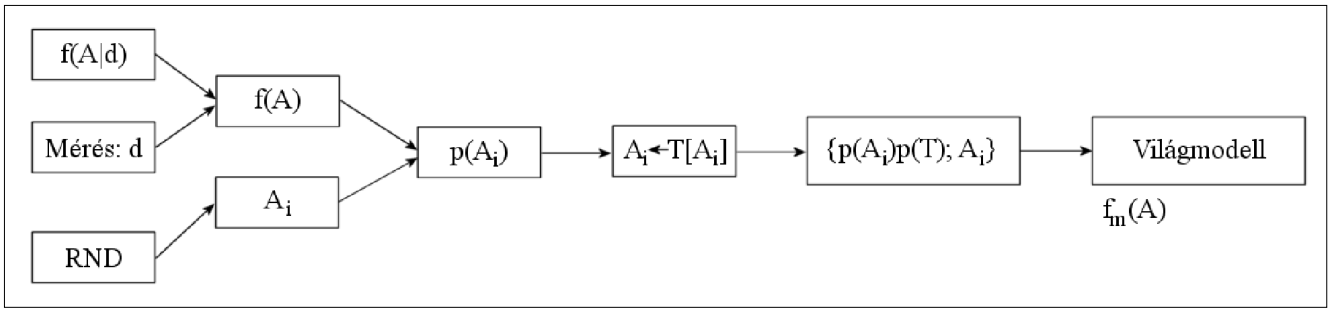
Már csak egy probléma maradt megoldatlan: a helymeghatározás. Adott ugyanis egy robot, ami csak távolságmérőkkel van felszerelve, és mint ilyen, fogalma sincs arról, hogy éppen merre lehet a világmodelljében, sőt, még azt sem tudja, hogy milyen irányba néz. Ezt hívjuk a lokalizáció (helymeghatározás) problémájának [3]. Robotunk blokkvázlata az *1. ábrán* látható.

Természetesen, az elmúlt években több megoldás is született ebben a kérdéskörben is, amelyek közül terjedelmi okokból csak a két leggyakoribb kategóriát emeljük ki: az úgynevezett landmarkos, illetve a térkép- és modellillesztő eljárásokét. Az előbbiek a szenzorjelek között keresnek a megszokottól eltérő, kiugró, csak a környezet adott, kis részeire jellemző mintákat (landmarkokat), míg az utóbbiak az aktuális szenzorjeleket transzformálva általában valamilyen mintaillesztő algoritmussal próbálják kitalálni az aktuális helyet és irányt.

A következőkben bemutatott eljárás is ez utóbbi kategóriába tartozik.

1. ábra Egy általános autonóm jármű blokkvázlata





2. ábra A világmodell felépítése

2. Részecskesereges optimalizálás

A lokalizáció feladata egy olyan T transzformáció „kitalálása", amely a robot lokális koordinátarendszerét egy globálisba viszi át. Kétdimenziós mozgást feltételezve ez egy 2D-s vektorral való eltolást, illetve egy elforgatást jelent. Ezen a háromdimenziós keresési teren kívül adott egy úgynevezett jósági függvény, ami megmutatja, hogy melyik transzformáció mennyire felel meg az elvárásainknak. Tehát a keresési terünk minden pontja egy lehetséges eltolás-elforgatás párost ad meg, amelyeket a jósági függvény segítségével ki tudunk értékelni.

A lokalizáció legegyszerűbb megoldása az, ha véletlenszerűen kiválasztunk néhány pontot a keresési térben, majd azokat kiértékelve egyszerűen a legjobbat használjuk fel a T transzformációként. Ennél egy fokkal hatékonyabb módszer az, ha ezeket a pontokat olyan részecskéknek tekintjük, amelyek mozogni is tudnak. Mozgassuk a kiértékelő lépés után az összes részecskét a legjobb jósági értékű pont felé! Néhány ilyen kiértékelés-mozgatás páros elvégzése után elvileg az összes részecskénk a legjobb transzformáció környékén fog tartózkodni.

Azért, hogy a pontjaink ne csak egy lokális legjobb érték köré álljanak be, hanem a globális optimumot találják meg, a pontok mozgatásán még kicsit finomítani kell. Így jutunk el a cikkben leírt lokalizációs algoritmus alapját képező Particle Swarm Optimization-ig (PSO, kb. „optimalizálás részecskesereggel"), amely alapvetően egy globális optimumkeresésre kitalált eljárás [2].

Ezt az algoritmust a kilencvenes években dolgozták ki úgy, hogy a kutatók az élelmet kereső madárrajok mozgását próbálták leutánozni. Jelenlegi formájában a részecskék („madarak") elmozdulásvektorát leíró egyenlet a következőképpen néz ki ($\vec{x}$ a keresési tér egy pontja):

$$\vec{v} = w \cdot \vec{v} + 2 \cdot RND_1 \cdot (\vec{P} - \vec{x}) + 2 \cdot RND_2 \cdot (\vec{G} - \vec{x}) \qquad (1)$$

RND_1 és RND_2 véletlen számok a (0...1)-ból, $\vec{P}$ az adott részecske eddigi legjobb állapotát (personal best) tárolja, míg $\vec{G}$-be az egész raj legjobb $\vec{P}$ értékét tesszük. A tapasztalatok alapján a konvergencia biztosítható akkor is, ha *w=1*, de ekkor az elmozdulásvektor hosszát maximalizálni kell.

Ha egy véletlen zajt adunk a $\vec{v}$-hoz, akkor a teljes beállás helyett a részecskék egy kis térrészben fognak mozogni az optimum körül. A (2)-ben ez a harmadik tag jelentősen elősegíti a gyorsabb beállást, ugyanakkor növeli is a lokalizáció hibáját. Szerencsére ez a zaj viszonylag nagy frekvenciájú, sokkal „gyorsabb", mint a robot mozgása, ezért egy egyszerű exponenciális szűrővel hatékonyan eltüntethető.

$$\vec{v} = \vec{v} + 2.0 \cdot RND_1 \cdot (\vec{P} - \vec{x}) + 2.0 \cdot RND_2 \cdot (\vec{G} - \vec{x}) + 0.5 \cdot RND_3 \cdot (\vec{R} - \vec{x}) \qquad (2)$$

Így sikerült elérni azt, hogy a PSO minden egyes számítási lépésben kis hibájú (tehát használható) eredményt ad. Ez azért lehetséges, mert egyrészt a részecskék sebessége jóval nagyobb, mint a roboté, másrészt a robot elmozdulása két lépés között kisebb, mint a részecskék által kitöltött térrész mérete.

A fentebb leírt $\vec{G}$ és $\vec{P}$ értékek a részecskék memóriáját jelentik, amelyek a környezetről tartalmaznak implicit információkat. Ezért, ha a robot mozog, ezeket időnként törölni kell.

3. A világmodell

A korábbi kutatások többnyire felülnézeti, valószínűségi térképet (gridet, rácsot), illetve gráfot használtak. Az előbbi túlságosan nagy tárterületet igényel, és a hosszútávú konzisztenciájának biztosítása is nehéz feladat – cserébe viszont könnyen felépíthető a szenzorjelekből. A gráfok ezzel szemben magasabb szintű információkat tartalmaznak, ezért rugalmatlanabbak, nehezebben felépíthetőek, viszont kevesebb memóriát igényelnek, és a navigációt is jelentősen megkönnyíthetik.

Mivel a fentebb leírt algoritmus egy alacsonyszintű eljárás, ezért egy alacsonyszintű világmodellt lenne célszerű hozzá használni: az akadályok valószínűségsűrűség-függvényét fogjuk véletlenszerűen mintavételezve lementeni *(2. ábra)*.

A robot távolságmérő szenzorai valószínűségi működésűek, ezért a kimeneti jelet szintén egy *a priori* valószínűségsűrűségi függvény adja meg. Ez megmutatja, hogy adott mért távolság *(d)* mellett melyik térrész (*A* terület) milyen valószínűséggel tartalmaz akadályokat. Jelöljük ezt $f(A|d)$-vel! A *d* ismeretében ebből meghatározható $f(A)$. Ezután kiválasztunk néhány véletlenszerű A_i térrészt (P pontokat és ε sugarú környezetüket), majd ezeken a területeken egy integrálással kiszámítjuk, hogy mekkora valószínűséggel lehet akadály:

$$p(A_i) = \int_{A_i} f(A) \cdot dA$$

Ekkor az A_i térrész még a robot lokális koordinátarendszerében van, ezért ezt a T segítségével áttesszük a globálisba. A világmodellben tehát eltároljuk a transzformált A_i térrészt, és annak $p(A_i)*p(T)$ „valószínűségét”, ahol $p(T)$ a kipróbált transzformáció jóságát, „valószínűségét” adja meg.

Ha $f(A|d)$ viszonylag egyszerű (például lézeres távolságmérő), akkor egy, esetleg két pont elég mérésenként, míg bonyolultabb függvény esetén többre is szükség lehet. Természetesen, minél több pontot használunk, annál pontosabb, stabilabb lesz az algoritmus, de a számításigénye is ezzel arányosan nő.

A világmodellt azonban nem csak építeni, hanem felhasználni is tudni kell. A PSO esetében ez csak annyit jelent, hogy a részecskék állapotát ki kell értékelni – így kapjuk meg a P-ket és a G-t a sok részecske helyzetéből. Szerencsére, a világmodell egyszerűsége miatt ez sem nagyon bonyolult feladat.

Mindössze annyi a dolgunk, hogy a világmodellt és az aktuális érzékelések valószínűségsűrűség-függvényeit azonos koordinátarendszerbe transzformáljuk, majd a világmodell ismert térrészeiben kiszámítjuk a szenzorjelek értékeit is. A kettő különbsége lesz a kipróbált transzformáció hibája *(3. ábra)*. Ezeket a különbségeket átlagolva kapjuk a kipróbált transzformáció jóságát, „valószínűségét”, *p(Ti)*-t.

4. Fejlesztési lehetőségek

Az eljárás jelenlegi formájában csak rövidtávon alkalmazható. Ez a valószínűségi működés miatt van így, ugyanis a világmodellben található akadálypontokhoz rendelt valószínűségek a kezdőponttól távolodva fokozatosan csökkennek. Ennek megoldására egy magasabb szintű algoritmusra is szükség van.

A beállást tovább lehetne gyorsítani, vagy, ami ezzel azonos értékű, a részecskeszámot csökkenteni, ha a robot fizikai modelljének részleteit figyelembe vennénk az eljárásban. Ezzel azonban vigyázni kell, hiszen minél több elemet veszünk be, annál rugalmatlanabbá válik az algoritmus.

Ha a T összeállításánál a G helyett a legjobb néhány részecske P értékét vennénk figyelembe, akkor valószínűleg a részecskeszámot tovább lehetne csökkenteni.

5. Összefoglalás

A fentebb bemutatott eljárás még jelenlegi, kezdetleges formájában is alkalmas az alacsonyszintű lokalizációra, mivel a szimulációk szerint nagyjából huszonöt centiméteres hibával képes volt a jármű helyét meghatározni.

A kutatás során a PSO-t a hozzáadott zajjal, szűréssel egészítettem ki, majd az algoritmust az új világmodellen futtattam, és így a tapasztalatok szerint egy jó kiindulási alapot kaptam egy teljes körű lokalizációs eljárás felépítéséhez. A működés pontosabb leírásához azonban egy jobb matematikai modellre van szükség.

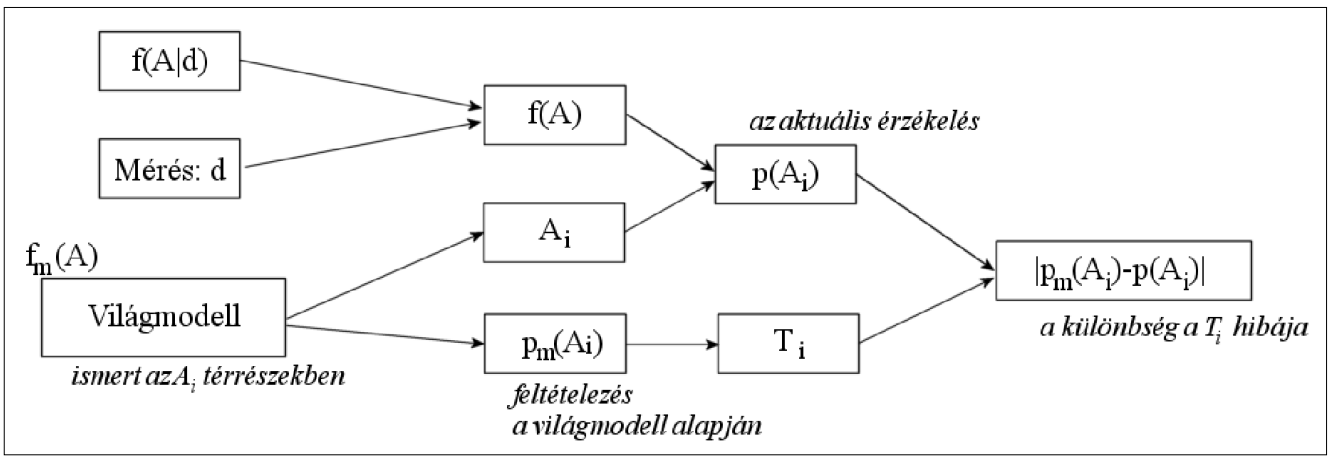
Irodalom

[1] Carlisle, A., Dozier, G.:
“Adapting particle swarm optimization to dynamic environments”, Proceedings of International Conference on Artificial Intelligence (ICAI) 2000, Las Vegas, Nevada, USA. pp.429–434.

[2] Eberhart, R. C., Kennedy, J.:
“A new optimizer using particle swarm theory”, Proceedings of the 6th International Symposium on Micromachine and Human Science, Nagoya, Japan. pp.39–43, 1995.

[3] U. Gerecke, N. E. Sharkey, A. J. C. Sharkey:
“Common evidence vectors for self-organized ensemble localization”, Elsevier Neurocomputing, October 2003, Vol. 55, iss.3, pp.499–519.

3. ábra Egy transzformáció (T_i) kiértékelése



A konferencia a digitális technika és az információ technológia egyre szélesebb körű alkalmazása következtében forradalmi változásokat megélő tartalom-előállító és tartalom-továbbító szolgáltatások legfrissebb technikai és technológiai megoldásait kívánja bemutatni, felvillantva a közeljövő nem kevésbé izgalmas perspektíváit is.
A konferencia egyaránt számít a tartalom-szolgáltatásban és a tartalom-továbbításban tevékenykedő mérnökök és technikusok, az e szakterületek felé orientálódó egyetemi hallgatók, de az e területeken kreatív vagy tartalmi, gazdasági, szervező munkát végző, nem technikai képzettségű menedzserek érdeklődésére is.

A konferencia fővédnöke:
Kovács Kálmán Informatikai és Hírközlési Miniszter

Fő támogató:
Antenna Hungária Rt.

A részletes program és jelentkezési lap megtalálható a www.hte.hu honlapon.
Kérhető továbbá a HTE Titkárságán
Tel.: 353-1027; fax: 353-0451; e-mail: hte@mtesz.hu
Postacím: 1055 Budapest, Kossuth tér 6-8.

Amennyiben hallgatóként kíván részt venni a rendezvényen, kérjük, hogy a jelentkezési lapot kitöltve szíveskedjen visszaküldeni 2005. május 18-ig a HTE Titkárságra.

Részvételi díj

A konferencia részvételi díja
64.000 Ft+ÁFA/fő/2 nap,
mely összeg magában foglalja az alábbiakat:

- szekciókon való részvétel;
- konferencia kiadványának egy példánya;
- résztvevők listájának egy példánya;
- kávészüneteken és ebédeken való részvétel.

Egyéni HTE tagok részére megpályázható kedvezményes részvételi díj:
48.000 Ft+ÁFA/fő/2 nap.

Feltételek:
- legalább 1 éves érvényes HTE tagság
- nincs tagdíjelmaradás

Kiállítás

A kiállításon épített kiállítási stand
és építetlen kiállítási terület igényelhető.

Kiállítási árak 2 napra:
- bútorozott kiállítási stand: 50.000 Ft/nm + ÁFA
- építetlen kiállítási terület: 35.000 Ft/nm + ÁFA

A kérhető minimális kiállítási terület 6 nm.

A standok elrendezését a szervező bizottság végzi.
A kiállítás területén éjszakai őrzést biztosítunk,
de nappal minden kiállító maga felelős
a standján elhelyezett tárgyakért, értékekért.

Amennyiben kiállítóként kíván részt venni a rendezvényen, kérjük, hogy a jelentkezési lapot kitöltve szíveskedjen visszaküldeni legkésőbb 2005. május 6-ig.

Kiadvány

A konferencián elhangzó előadásokat
kiadványban jelentetjük meg,
melyet a résztvevők a regisztrációnál kapnak meg.
A kiadványban lehetőséget kínálunk
hirdetés megjelentetésére, A4 oldal méretben
25.000 Ft+ÁFA díjért.
A hirdetéseket fehér papírra, jó minőségű
fekete-fehér nyomtatásban legkésőbb 2005. május 6-ig
kérjük eljuttatni a HTE Titkárságra.

Várjuk jelentkezését!

A 11. Televízió- és hangtechnikai konferencia és kiállítás
Szervező Bizottsága

Elindult a megújult „Biztostű”

HORNÁK ZOLTÁN

hornak@mit.bme.hu

Az ELTE és a BME közös erőfeszítéseként 2003. augusztusában elindult a Biztostű informatikai biztonságtechnikai oktató portál, amelynek célja, hogy a téma iránt érdeklődő egyetemistáknak és szakembereknek nyújtson segítséget az eligazodásban. A portálon egyaránt megtalálhatóak a biztonságtechnika általános alapelvei és konkrét technikái, így a Biztostű nem csak a biztonsági szemléletmód megalapozásában segít, hanem a későbbi szakemberek is kiválóan használhatják referencia anyagként.

2004-ben két további pályázat anyagi támogatásával a BME és a SEARCH-LAB Kft. a honlapot formailag és tartalmilag is teljesen megújította. A meglevő tartalmat felülvizsgálták és kiegészítették, és azt a MOODLE szabad forrású e-learning keretrendszerbe ültették át. Elkészült továbbá a tartalom két, egyszerűsített változata is: a MOODLE segítségével tananyagokat, komplett tanuló utakat alakítottak ki mind a középiskolai szint, mind az általános iskolai szint, vagy alapfokú érdeklődéssel rendelkezők számára. A „tananyag” elsajátítását, önellenőrzését minden fejezet után beépített és automatikusan kiértékelt tesztek segítik.

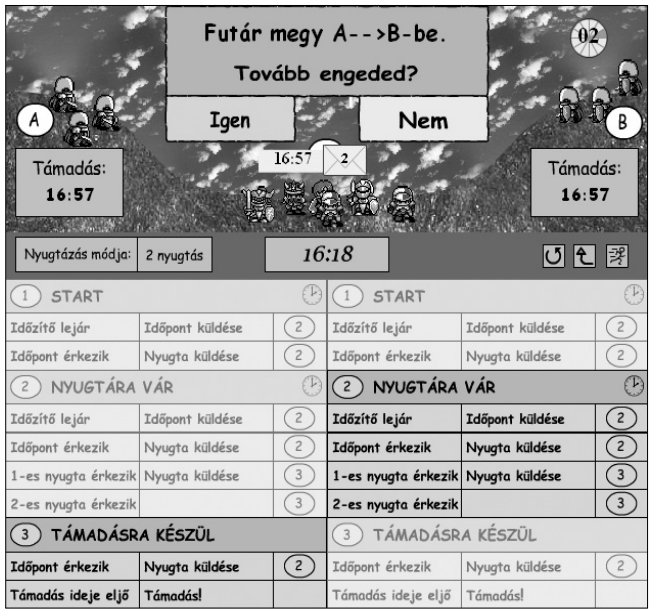
A már korábban is meglevő három játék az új rendszerben további hárommal bővült. A tananyaghoz kapcsolódó Flash alapú játékok szemléletessé teszik a biztonsági problémákat, segítik az alapelvek mélyebb megértését.

A nagysikerű jelszó kitaláló játék a tipikus jelszó-törő programok működését szimulálva, magyar nyelvű szavakat is tartalmazó szótár alapján demonstrálja, hogy a majd mindegyikünk által használt jelszavak miért és mennyire gyengék, és végigvezeti a felhasználót az erős jelszavak tudományának rögös útján *(1. ábra)*.

1. ábra

A bizánci játék azt a kommunikációs protokollokban fellelhető problémát szemlélteti, hogy két fél között a biztonságos kommunikáció bizonyos körülmények között mindig meghiúsítható *(2. ábra)*.

2. ábra



A játékok a használatához szükséges tippeken felül a mögöttes elvek megértését segítő magyarázatok is elérhetők az oldalon.

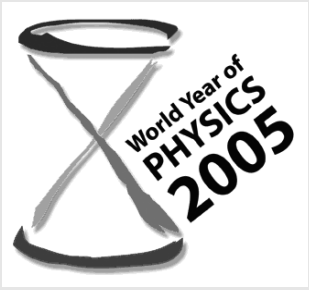
A portál rendkívül érdekes és hasznos része az „ökölszabályok” gyűjteménye, ahol az informatikai biztonság sarokpontjait, alapigazságait sulykolják az oldal készítői a látogatóba. A tananyagokban a látogatók érdeklődési szintjüknek megfelelően kereshetnek, de a készítők fogalomkeresővel is kiegészítették a már meglevő rendszert.

Az új rendszer 2004. őszi indulása óta már közel hétszáz regisztrált felhasználója van a Biztostűnek, aminek egy részét a BME-n illetve az ELTÉ-n tanuló biztonság szakirányú hallgatók teszik ki.

A honlap címe:
www.biztostu.hu

2005: a Fizika Nemzetközi Éve

SIPOS LÁSZLÓ
siposlaj@axelero.hu



2000 év végén – a Fizikai Társulatok Világkonferenciáján – több mint negyven társulat támogatta azt a javaslatot, hogy 2005-öt nyilvánítsák a Fizika Nemzetközi Évének, majd a UNESCO is felkarolta a kezdeményezést. A javaslattevők célja, hogy ez az év hozzájáruljon a fizika és szélesebb értelemben a természettudományok társadalmi presztizsének javításához. Érdemes megvizsgálnunk, hogyan hatott a fizika a társadalom, a kultúra és gondolkodásunk fejlődésére. A fizika hatása közvetlen módon jelenik meg hétköznapi eszközeinkben, és meglepően sokféle módon járulva hozzá életminőségünk javításához.

Albert Einstein (1979-1955), a berni szabadalmi hivatal műszaki szakértője 1905-ben több – ma már tudjuk, hogy fizikatörténeti jelentőségű – cikket közölt az Annelen Physik című folyóiratban. Ezekben magyarázatot adott a fényelektromos jelenségre, és bevezette a foton fogalmát; értelmezte a „nyugvó folyadékban lebegő részecskéknek a hő molekuláris elméletéből következő mozgását"; megalapozta a speciális relativitáselméletet; levezette a tömeg és az energia ekvivalenciáját kifejező $E=mc^2$ összefüggést. A dolgozatok közül a kvantum-hipotézis, illetve a speciális relativitáselmélet a nem szakemberek számára is a modern fizika szimbólumává vált. Így érthető, hogy Einstein „csodaévének" centenáriumát választották az ünnepi évnek.

A Fizika Nemzetközi Évében világszerte és idehaza is számos rendezvényt terveznek, melyek célja, hogy felkeltsék a fizika iránti érdeklődést és hozzájáruljanak a fizika – szélesebb értelemben véve a műszaki-, és természettudományok – társadalmi elfogadottságának növeléséhez. Az Év hazai indító eseménye január 11-én, a Magyar Tudományos Akadémia épületében tartott sajtótájékoztató volt. *„Az elmúlt száz év a fizikáé volt, de legalább ennyire a hazai fizikusoké is, akik ma is ott vannak a világ legnevesebb intézményeiben, öregbítve a magyar tudomány jó hírét"* – kezdte mondandóját Vizi E. Szilveszter, az MTA elnöke.

„Nemcsak elméleti szinten lépett nagyokat a fizika a huszadik században, hanem behatolt a mindennapjainkba: se szeri, se száma azoknak a hétköznapi és speciális eszközöknek, amelyek a fizika gyakorlati alkalmazásai." – jelezte Németh Judit, az Eötvös Loránd Fizikai Társulat elnöke. Megtudtuk, hogy a Fizikai Szemlében új rovat indul, amelyben diákok, tanárok, fizikusok közölhetik legújabb eredményeiket. A társaság területi- és szakcsoportjai egész évben folyamatosan szerveznek filmvetítéssel egybekötött előadásokat, többek között a Műegyetemen, az ELTÉ-n és a Csodák Palotájában.

„3+1 filmmel készülünk az évre" – jelentette be Bencze Gyula, a Magyar Mozgókép Közalapítvány tudományos filmműhelyének vezetője. Az első alkotás egy portréfilm, amely Simonyi Károllyal készült, a második „A fizika kultúrtörténete" címet kapta, a harmadik pedig „Ősrobbanás" néven kerülhet vászonra. Utóbbinak külön érdekessége, hogy a filmet készítő forgatócsoport bejutott a világ legismertebb svájci részecskegyorsító intézetébe, ami „kamerás embereknek eddig még nem sikerült". A negyedik, most készülő alkotás („A szegedi lézeresek") egy riportfilm a Bor Zsolt és Szabó Gábor neve fémjelezte lézerfizika-kutatócsapat hétköznapjairól.

Egyed László, a Csodák Palotájának vezetője szavai szerint intézményükben annak alapítása óta zajlik a „fizika éve". Kiemelte, hogy az Öveges-teremben rendszeresen élőben zajlanak a kísérletek. Erre a kezdeményezésre épül majd a Csodák Palotájába szervezett jubileumi rendezvénysorozat egyik legfontosabb eleme, az otthon kísérletező fizikatanárok bemutatkozása.

Az európai fizikusok kezdeményezésére vándorkiállítás indul a világon – tudtuk meg Nagy Dénes Lajostól, az 1968-ban alakult Európai Fizikai Társulat (European Physical Society) Konferencia Bizottságának elnökétől. A nemzetközi évnek további apropója, hogy 50 éve, 1955-ben hunyt el Einstein Princetonban. Halálának évfordulóján, április 18-án az amerikai egyetemi városban kialudtak a fények, majd kisvártatva egy fényjel indult útjára, hogy staféta formájában körbejárja a Földet és Princetonba visszatérve újból fényár borítsa a tudós egykori lakhelyét. Az akcióhoz Magyarország is csatlakozott, így április 19-én Románia és Szerbia irányából hozzánk is elért a fénystaféta, amelyhez autófényszóróval, tábortűzzel, zseblámpával, de akár engedélyezett tűzijátékkal is bárki kapcsolódhat.

Az idén 136 éves Természet Világa folyóirat számos cikkfolyamban foglalkozik a fizika tudományával – számolt be terveikről Staar Gyula, a lap főszerkesztője.

Szilágyi Zsuzsa a Mindentudás Egyeteme (ME) kapcsolódó eseményeiről számolt be. Szabó Gábor nyitóelőadását további négy fizikai tárgyú előadás követi: Kolláth Zoltán a csillagbelső hangjairól, Faigel Gyula az atomi szerkezetekről, Fodor Zoltán a világegyetem keletkezéséről, Kroó Norbert szemeszterzáró előadása pedig a fény fizikájáról szól.

Vizi E. Szilveszter zárszavában kiemelte, 2005-ben nagyon is időszerű az áttörés a természettudományok ismeretterjesztésében, hiszen egyelőre ott tartunk, hogy a hazai sajtótermékek még a világszenzációt jelentő publikációk mellett is gyakran szó nélkül elmennek.

K+F?! Kutatás?! Innováció?!
Interjú Havass Miklós informatikussal, a Számalk Rt. elnökével

NAGY BEATRIX HAVASKA
nbh@vipmail.hu

2005. január 27-én került sor a Nemzeti Fejlesztési Hivatal felkérésére a „Magyarország jövőképe" sorozatban egy, az innováció témakörében megrendezett beszélgetésre, melyen Havass Miklós tartotta a bevezető előadást. Előadása keretében elhangzott tőle tíz, a kutatás-fejlesztés jövőjével kapcsolatos kérdés:

1. Mit ígérhetne a kutatói szféra, ha a Nemzeti Fejlesztési Tervben tisztességes támogatást kapna? Hogyan, mennyi térülne meg az invesztícióból? 2. Ki ígérheti a visszatérülést, megfelelő-e a struktúránk ahhoz, hogy valaki felelősséggel ígérhessen? 3. Hol, kiknél fognak hasznosulni a K+F eredményei? (multiknál, magyar kis-, és közép vállalatoknál?) Mennyi a hasznosulás realitása? 4. Netalán az állam szervezi úgy egyéb tevékenységet, hogy a K+F ottani feladatokat old meg? 5. Mi a K+F tevékenységek szelekciós kritériuma? 6. Milyen módon visszük az eredményeket gyakorlatba? Van erre általános kultúránk, ezt segítő bürokráciánk? 7. Milyen lesz a nemzetközi beágyazottság? (Nem fogják-e az agyat külföldről elszívni? Hogyan terjesztjük ki a K+F piacunkat egész Európába? Hogyan dolgozunk együtt velük?) 8. Hogyan csökkenthetjük le az akadályozó bürokrata korlátokat és gyorsíthatjuk fel a folyamatokat? 9. Akarunk-e egy európai méretű K+F intézetet? Milyen területen? 10. Hogyan kapcsolódik be alkotó, hatékony és felelősségteljes módon a vezetőértelmiség?

Ezek kapcsán felmerült; vajon hogyan válaszolná meg a saját kérdéseit?

Hogyan befolyásolja a Nemzeti Fejlesztési Terv az ország kutatásának, gazdaságának, fejlődésének irányát, ad-e valami többletet a kutatási szféra számára, illetve hatással van-e az innovatív magyar gondolkodásra?

Folyik a Nemzeti Fejlesztési Terv készítése, amelynek során át kell gondolnunk azt, hogy 2007-és 2013 között milyen strukturális változtatásokat vezessünk be az országban ahhoz, hogy jöjjön létre egy olyan hely, amelyik közel áll Európához, amelyik tetszik nekünk, s amelyikben élni szeretnénk. Ám a tervezők zöme úgy gondolja, hogy Magyarország számára egy olyan jövőt kellene elképzelni, amelyikben kiemelt szerepe van földrajzi környezetünknek.

Bennünket olyan államok vesznek körül, amelyek még nem csatlakoztak, de csatlakozni fognak az Európai Unióhoz. Ez számunkra egy olyan tranzit szerepet biztosíthat, mellyel akár Európa egy térségének intellektuális centrumává is válhatunk. Ezt akkor tudjuk megvalósítani, ha ebben a fejlesztési periódusban olyan beruházásokat hajtunk végre, amelyek megerősítik azokat a kutatási potenciálokat, melyek segítségével létre tudjuk hozni a munkaerőnek, a kapacitásoknak egyféle regionális koncentrálódását, cizellálódását, amely természetesen kétirányú, és egy állandó cserekapcsolatot is indukál.

Ahhoz azonban, hogy például Magyarországon felépítsünk egy fontos európai kutató intézetet, amely állományának egy részét a régióból toborozza, ehhez vonzóvá kell válnunk, és akikre számítunk, például kutatókra, szívesen jöjjenek ide, szívesen éljenek itt. A vonzó ország fogalma alatt nagyon egyszerű dolgokat értünk. Mi is csak akkor települünk egy másik országba, ha biztos és jó egészségügyi ellátás van, jók a kórházak, van kultúrális kínálatuk, nincsenek napi közlekedési gondok, vannak útjaik, a kutatók a gyerekeiket jó helyen tudják taníttatni, mert jó oktatási rendszerrel rendelkeznek. Egy vonzó hazakép alkalmas arra, hogy az általunk magasan képzett és külföldön is jól elhelyezkedni tudók szívesen itt maradjanak, és további kutatók is jöjjenek hozzánk.

Az elképzelés megvalósítható, mert népünk általában ismert invenciós készségéről, az új kapcsolatokat felderítő képességéről, ennek ellenére még mindig vannak problémáink az elképzeléseknek a gyakorlatba ültetésével, megvalósításával. Jövőnk szempontjából ezért fontos, hogy megoldjuk annak a környezetnek a kiépítését, amely segíthet a kutatói munka eredményeképpen létrejövő új találmányok realizálódásában, gazdasági értékeket teremtve. Ehhez gondolkodnunk kell a tudomány-irányítási rendszerünkről is.

A tudományos kutatások alapját a Tudományos Akadémia biztosítja, amely történelmileg régi, patinás centruma az alapkutatásoknak. A második világháborútól kezdve azonban Amerikában a szokásos természettudományok mellett a tudománynak egy új ága hajtott ki: a technológia.

A technológia a természettudományos eredmények megtestesülése a gazdasági folyamatokban. A technológiáknál konkrét feladatokat kell határidőre megoldani, és ezeknek bizonyos megtérülésekkel kell járniuk. Ez egy másfajta tudomány-szervezést kíván.

Ha egy jövőbeli Magyarországra gondolunk, amelyben központi szerepet szánunk az innovatív magyar gondolkodásnak, akkor világosan kell látnunk, hogy mind a kétfajta tudományra szükségünk van: a világot megismerő alap tudományokra, ezen belül természettudományokra, amelyeknek élniük kell a saját kritérium

rendszerük szerint; és szükségünk van a technológiára, melynek irányítása, szabályozása új módszereket igényel, amelyet az NKTH-nak kell megtalálnia.

Melyek a szelekciós kritériumok? Hogyan hasznosulnak a K+F eredmények?

Egy kutatás szükségességét formális kritériumok szerint választjuk ki, nem a téma szépsége alapján, hanem aszerint, hogy abban a magyar környezetben, amelyben élünk, és amelyben gazdálkodó egységek, iparszervezetek dolgoznak, a várható eredménynek lesz-e gazdasági haszna, születhet-e termék egy-egy folyamat végén.

Ma a kiválóan képzett magyar szellemet sokszor közvetlenül a munkáján keresztül értékesítjük. Ennek az elképzelésnek az a hátránya, hogy mindig csak az egyszeri munkát fizetik meg, így felhalmozott többlet értéket nehéz megteremteni. Jelentős mennyiségű haszon eléréséhez arra van szükség, hogy az elért eredményeket többszörösen tudjuk felhasználni, értékesíteni, ami egyenértékű azzal, hogy a kutatás-fejlesztési munkák végére eladható piaci termék jöjjön létre, amelyet a világpiacon is tudunk értékesíteni. A termék piacra vitelének marketing, PR, jogi követelményei vannak, ezért a saját jog- és szabályozó rendszerünket úgy kell átalakítani, hogy ezen termékeknek a létrehozása minél könnyebb lehessen.

Az újdonságok először természetesen a magyar piacon kerülnek eladásra. Itt mérik be őket, megfelelnek-e a felhasználhatóság kritériumainak. Igazán jelentős egy termék, ha nemzetközi piacra tud kerülni. Vagyis a technológiai jellegű kutatás-fejlesztések minősítésének az igazi kérdése az, hogy, a folyamat során előállított termék megfelel-e a nemzetközi kritériumoknak, illetve helyt tud-e állni a nemzetközi versenyben is.

Milyen és mekkora az állam szerepe ebben a folyamatban?

Egy kutatás fejlesztési rendszer kiépítésénél mindig nagyon fontos, hogy egy tőkeszegény országban, mint amilyen Magyarország, milyen legyen az állam szerepe. Itt nagy kérdés, hogy vannak-e olyan prioritások, amely a magyar jövő, a magyar gazdaság és ennek kapcsán a magyar kutatás-fejlesztés számára különösképpen fontosak és ajánlottak.

Erről napjainkban nagy vita folyik. Egy azonban biztos; hogy koncentrálni kell bizonyos területekre, ahol nemzetközi szinten átütő eredményeket tudunk elérni. Magyarán szólva prioritásokra, méghozzá kevés prioritásra van szükség, amelyekben átütő eredményeket érhetünk el.

Hogyan jelöljük ki ezeket a prioritásokat?

Ez bizony nehéz kérdés. Nyilvánvalóan ki lehet abból is indulni, hogy milyen vállalataink vannak, milyen területeken folyik termelés, hova megy az export, hol lesznek szükségesek újabb és újabb fejlesztések ahhoz, hogy a piacon létünket, exportképességünket növelni tudjuk.

Kiindulhatunk abból is, hogy mihez vannak különleges adottságaink, hol vannak olyan iskolák, szellemi közösségek, melyektől már az eddig elért eredményeik alapján remélhető, hogy nemzetközi szinten is átütő eredményeket tudnak elérni. De kiindulhatunk abból is, hogy Magyarországnak bizonyos területeken nagyon nagy elmaradásai, és az európai környezethez képest kirívó problémái vannak: környezetszennyezés, csatornázás, szennyvízkezelés, egészségügyi ellátás, az urbanizáció, a falusi életmód, a mezőgazdaság átállása és modernizálása.

Ezeket a kérdéseket meg kell oldani. A megoldásához koncentrált befektetés szükséges. Miért ne választhatnánk a kutatás-fejlesztési prioritásokat ebből a szükségből. Találjunk ki olyan megoldásokat és invenciókat, amelyekkel ezeket a problémákat könnyebben oldjuk meg. Ez hasznos, mert ezeknek a talaján, a világon más olyan régiókban is, ahol szintén ilyen problémákat kell megoldani, termékszállító kész állapotba tudunk kerülni.

Annak idején például a hollandoknak kifejezett hátránya volt a kevés földterületük. Meg kellett küzdeni a természettel azért, hogy több földterületet hódítsanak el, a tengertől, ezért nagy teherbírású gátakat építettek. Ezzel olyan készséget fejlesztettek ki, hogy ha ma gátakat, vagy más hasonló nagy természeti létesítményeket kell a világon bárhol létrehozni, akkor általában a holland szakembereket és módszereket alkalmazzák. Ugyanígy hiszem azt is, hogy Magyarországon egy-két jó prioritással meg tudjuk segíteni a hátrányaink csökkenését, ugyanakkor be tudunk törni a nemzetközi piacra.

A magyar bürokrácia akadályozza vagy segíti a folyamatokat? A kutatói réteg ígérheti-e a befektetések megtérülését?

A Nemzeti Fejlesztési terv fő célja a magyar versenyképesség növelése és ennek következményeként a magyar élet minőségének javítása. A bürokratikus államigazgatási, közigazgatási környezetnek nem akadályozni, hanem elősegíteni kell ezt a folyamatot. A mai modern, globális világgazdaságban, egy-egy verseny megnyerése vagy jó szereplés egy-egy versenyen, sokszor már vagy nemcsak a termék minőségén vagy a termék fejlesztésében résztvevő emberek kvalitásán múlik, hanem azon a környezeten is, amely a termék elterjedését gátolja, vagy elősegíti.

Ilyen értelemben a magyar államigazgatás nem egyszerűen kellemes vagy kellemetlen környezet a jövő szempontjából, hanem versenyképességünk egyik alapvető feltétele, amiből az következik, hogy ezt – a Bach-korszakban, tehát az elvesztett szabadságharc után uralkodó korszakban kifejlesztett porosz mentalitással létrejött, olykor nagyon hasznos, rendet teremtő, de körülményes államigazgatást – át tudjuk-e egy versenyt segítő, rugalmas, modern államigazgatássá alakítani.

A magyar állam kétféle értelemben sem támogatja eléggé a kutatás-fejlesztést. Talán nem szándékai szerint, hanem mert történelmileg így alakult. Az egyik kérdés, hogy mennyi pénzt ad, a másik, hogy milyen feltételeket ad ahhoz, hogy az eredményes lehessen.

Szoktunk sírni azon – és ez általában igaz is – hogy a GDP-hez viszonyítva a magyar állam keveset ad kutatás-fejlesztés támogatására. Bár ez valóban probléma, azonban ez a kérdés is kétoldalú. Ha megnézzük a környező országokat, tehát azokat, akik ugyanebben a gazdasági nehéz felzárkózási periódusban vannak, akkor azt kell mondanunk nem kirívó a helyzetünk, a többiek sem kapnak sokkal többet. Egyszerűen a létezéshez sem jut elegendő pénz, így a jövőt előkészítő K+F-re sem. De ha többet is adna az államigazgatás, vajon tudná-e a K+F szféra garantálni, hogy ettől meg fog nőni a gazdaság teljesítménye? Ha ez ilyen egyértelmű lenne, akkor valószínűleg többet kapna az ágazat. De nem tudjuk biztosan ígérni, mert ez a játék többszereplős. Multinacionális cégek, magántulajdonú középvállalkozások, állami bürokrácia és a világpiac együttes játéka határozza meg azt, hogy egy befektetésből milyen eredmény születik.

Akkor emelhető a támogatás mértéke, ha olyan környezet jön létre, aminek segítségével a K+F-be befektetett pénz valóban hasznosulni tud, többletet tud termelni, amit újból be tudunk fektetni, például a K+F-be. Azt gondolom, hogy ez a felismerés már meg van. Az állam többet szeretne adni, az Európai Fejlesztési Terv során a prioritások közé tűzi azt, hogy az Európától kapott többletforrásokat elsősorban tudásigényes feladatokba fekteti. De ehhez szükséges a másik oldal vállalása is: „ha az Állam többet ad, és rendezi a környezetet, mi vállalkozók, vállalatok valóban produktívan fogjuk ezt felhasználni, és az államnak több eredményt produkálunk, amiből természetesen nekünk is több hasznunk lesz” – ez az értelme az igazi együttműködésnek, ahol mindketten jól járnak.

Én úgy érzem, hogy az Európai Terv tervezése folyamán nagy elszántsággal merül föl az, hogy Magyarország innovatív központtá váljon, így kiemelt szerepe legyen a kutatás-fejlesztésnek és annak a hasznosulásának. Most azt a környezetet kell megterveznünk és megvalósítanunk, ami ezt lehetővé is teszi.

Van-e lelkes, fiatal kutatói réteg Magyarországon, akik tovább viszik a nagy elődök eddig elért eredményeit?

Sok tehetség van ma is. Sokszor elhangzik, hogy az egyetemek színvonala, hallgatósága felhígult, a tanári kar kiöregedett, financiális problémák vannak. Ez mind igaz, de ebben a felhígult tömegben is sok nagyon tehetséges, ambiciózus, leleményes fiatal van. Ezek előbbutóbb elhagyják az iskolapadot. Ők már úgy nevelkednek fel, hogy már tanulmányi időszakaik alatt is nyelveket tanulnak, külföldre járnak, netalán külföldön hallgatnak bizonyos időszakokat, külföldi projektekben is részt vesznek.

A kérdés az, hogy ezek a külföldöt járt végzett ifjak megtalálják-e itthon az ambícióiknak megfelelő helyet, vagy sem? Hogy itthon kevesebbet keresnek-e vagy sem, vagy hogy mindaz, amit egy ország tud adni az állampolgárainak, kompenzálja-e a külföldön elérhető fizetéstöbbletet? Tudunk-e jó orvosokat, egészségügyi ellátást, oktatást, színvonalas kultúrát, szórakozást, jó lakásfeltételeket adni? Ha igen, akkor itt maradnak és úgy érzem nem lesz gond a fiatalokkal. Ha nem tudjuk ezt biztosítani, akkor az európai csatlakozás csak egy óriási lehetőség volt, hisz szabadságot adott a mobilitásra, de ez a lehetőség a visszájára fordul, mert éppen a legértelmesebb rétegünk vesz vándorbotot és itt hagy bennünket. Esetleg szellemileg kiürül Európának ez a része...

Sokszor dicsérjük Amerikát, akik a gazdasági verseny élén járnak. De vegyük észre, e mögött náluk hatalmas mobilitás áll. A keletről nagy tömegek mentek át a nyugati partvidékre, mert ott jobbak a munka és kutatás-fejlesztési körülmények. Ha ez Amerikán belül történt, akkor erre megvonjuk a vállunkat, és azt gondoljuk, hogy na és: egy országon belül vándoroltak. De ha arra gondolunk, hogy Európa kereteiben történik meg ugyanez, akkor nálunk előfordulhat az, hogy a kutatói réteg Magyarországról átvonul egy másik országba.

Európa szempontjából ez persze pozitív és hasznos is lehet. A koncentráció, a mobilitás a legfontosabb versenyfeltételekhez tartoznak. Nekünk, mint nemzetnek azonban ez tragédia lenne. Így amikor a jövőnket tervezzük, és azt mondjuk, hogy egy élhető, jó minőségű országot kell teremteni, akkor pont ezt szeretnénk elérni, hogy vonzóak legyünk. Még akkor is, ha nem tudjuk rögtön anyagi lehetőségeinkben utolérni a nyugatot.

Folytatva az előző gondolatkört az EU csatlakozás miatt kell-e félnünk az agyelszívás növekedésétől?

Régóta szembesültünk azzal, hogy néhány nagy rendszer struktúráját még nem modernizáltuk. Az államigazgatásunk, közigazgatásunk, egészségügyi ellátó rendszerünk például ilyenek. Ezek megváltoztatása sok pénzbe kerül, és óriási érdekek fűződnek egy-egy már kivívott pozíció, elosztási rendszer megtartásához, és ezek a rövidtávú kormányzati ciklusok nem merik vállalni ezeket a kihívásokat. Továbbá nehéz megoldani azért is, mert az emberi természet a járt utakat szeretné továbbra is járni.

Ennek ellenére most történelmi pillanat van, mert minden eddiginél több pénzt fogunk kapni az Európai Uniótól. Ehhez az kell, hogy a napi politikában szembenálló erők, a főbb kérdésekben egyetértsenek és együttesen vállalják azokat a kellemetlen következményeket és nehézségeket, amelyek a fennálló akadályok legyűréséhez szükségesek. Közös eltökéltséggel megtörténhet a jogszabályok, a környezet megváltoztatása. Ha ez nem következik be, akkor nem változik a környezet sem, de akkor bizony nem nézünk „szép jövő” elé...

Egy szép saját jövő, vagy egy Európában végződő végkifejlet között kell választanunk. Hiszek magunkban! Hisz éppen az európai kis országok között vannak olyan élő példák, ahol ezt a kihívást sikerrel válaszolták meg. Ahol a történelmi előzmények nem sok jóval kecsegtettek, és mégis – bátor vállalásokkal – nagyon szép jövőt teremtettek maguknak. Ilyenek például: a finnek, az írek, újabban az észtek. Gyönyörűen fejlődnek, mert valamit nagyon eltökélten végeznek. Ezt kell nekünk is tenni.

Az EU tagság előnyös-e a K+F számára? Hogyan tudunk megfelelni ezen a fejlett piacon?

Magyarországon a K+F-fel kapcsolatban van egy nehézségünk, aminek nem látom még világosan a megoldását, de valamiképpen ezt a dilemmát meg kell oldanunk.

A magyar gazdaság úgy lett Európa-konform, hogy sok jelentős multinacionális céget beengedtünk, sőt mi hívtunk Magyarországra. Kialakult egy multinacionális vezetésű ipar Magyarországon, amelyik megsokszorozta exportunkat és ezzel életképessé tett bennünket. A multinacionális cégek mögött, a magyar vállalkozások száma ugyan nagy volt, de tőkeereje csekély. A multinacionális cégek, azt, hogy mit, hol végeznek el, természetesen saját globális szempontjukból vizsgálják. Nem feltétlenül Magyarországra, helyezik ki kutatásaikat. Tehát a magyar telephelyű multinacionális ipar nem feltétlen erősíti a magyar K+F-et, vagy nem olyat, ami egyébként magyar szempontból előnyös, hasznos volna.

Döntő lehet az, hogy a magyar kis- és középvállalkozások felnőjenek és használják a magyar K+F eredményeit. Azonban a pillanatnyilag ezek a kis- és középvállalkozások napi piaci problémáik, jelenlegi alkalmazkodási készségük miatt még nem nyitottak erre. Többnyire világpiaci eredetű termékeket adnak tovább és keveset újítanak. Ma még súlyos kérdése a magyar K+F politikának, hogy hogyan lehetne ezt a szférát aktiválni. Szerencsére bizonyos multinacionális cégeknél már van néhány példa arra, hogy kutatási tevékenységük hasznosuló jelentőségét is Magyarországra összpontosítják.

Ebben a hatalmas gazdasági versenyben – akár az EU-n belül – meg tudjuk-e tartani az országra jellemző egyediségünket, vagy a globalizáció áldozatává válunk mi is?

Nagy dilemma ez pillanatnyilag. Minden értelmiségi tudja, hogy hatalmas világverseny részesei vagyunk, amely a saját gazdasági törvényei szerint folyik, s amely verseny nem feltétlenül támogatja a szép, emberi értékeket. Néha segíti, sokszor azonban elnyomja azokat. Tudnunk kell azonban azt, hogy Amerikában még ma is szívesebben élnek az emberek, mint valamelyik szegény Afrikai vagy Ázsiai országban. A gazdagság globálisan felkínál lehetőséget, de lépést kell tartani, ez a feltétele annak, hogy egyáltalán önálló gazdaságot, politikát tudjunk folytatni.

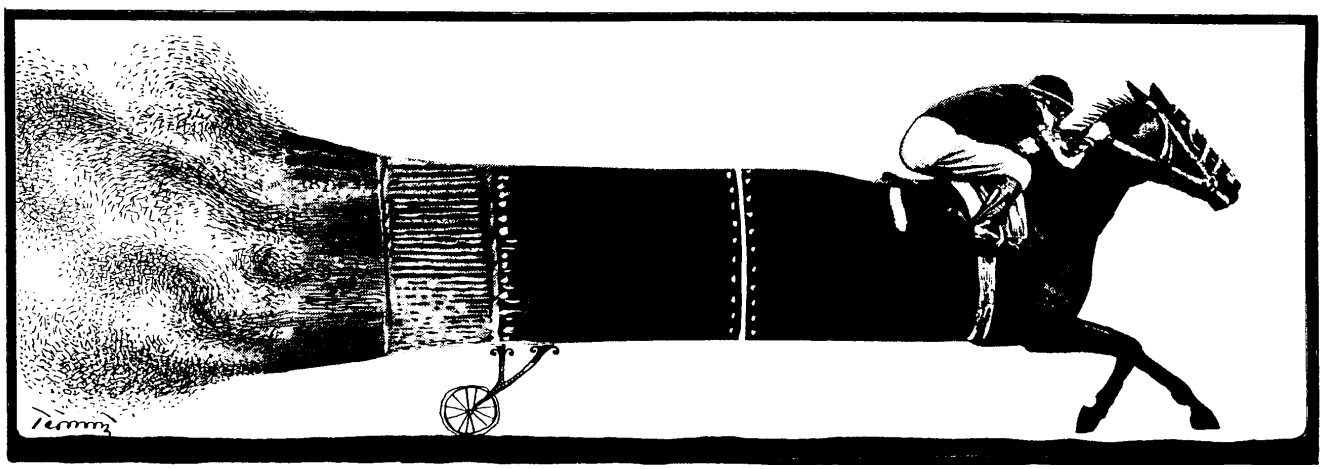
Az más kérdés, hogy amikor már megy a lépéstartás, hogyan alakulnak a viszonyok. Felhalmozódik egy sor probléma ezzel a szabadversenyes piaccal kapcsolatban és előbb-utóbb megszületnek az alternatív megoldások, válaszok is. Konkrétan a tömegtermelés nagyon fontos, ettől lesz valami nagyon olcsó, eladható; ettől lesz nagy jövedelem, és így tovább. A másik oldalon én, mint vásárlóképes egyén, egyre inkább az individuálist keresném. Azt, ami sajátosan nekem szól, ami egy kicsit különleges.

Nemrég tértem haza Marokkóból, ahol számomra kiábrándító volt, hogy ha elmentem egy üzletbe, egy nagyáruházba, akkor márkáról márkára ugyanazokat a csokoládékat, tejeket, kenyereket találtam, mint másutt, vagy akár itthon is. Holott én azért mentem oda, mert valami mást szerettem volna kapni. Az ember a saját életében keresi az individuálisat is. Meg fog születni az a kívánság, hogy sajátos termékek jöjjenek létre.

Szerencsénk van, hiszen az automatizálás, az információ-technológia jelenlegi foka elősegíti azt, hogy akár személyre tervezett termékek jöjjenek, nagy hatékonysággal létre, és ebben igenis lehet a kis országoknak is nagy szerepe. Lehet esetleg kutatás-fejlesztési kitörési pontokat találni Magyarország számára is. Ugyanígy végig lehet gondolni a kultúrális adottságokat is.

Azt gondoltuk, hogy az Internet elterjedésével megszűnnek a nemzeti nyelvek, mert óriási kényszerítő ereje van egy közös nyelv használatára. Furcsa és érdekes, hogy amióta Internet van, éppen a világhálón egyre drámaibb mértékben nő a sajátos, kis, esetleg már elfelejtett nyelvek használóinak a tábora. Azoké a kis közösségeké, akik igenis írül meg baszkul akarnak érintkezni, levelezni amellett, hogy tudják az angolt is.

Azt gondolom, hogy az ember alapvetően individuális lény. Legnagyobb kincse az, hogy megismételhetetlen, egyedi valaki, és ez az egyediség mindig keresni fogja a különbözőségének útjait.



SERVICE-PROOF COMPUTER TECHNOLOGY

Testing model of software systems

Key words: security-critical systems, verification, validation, error model, formal methods

This paper deals with general considerations of testing complex software systems with focus on the software of security-critical computer systems. It starts with software errors to be taken into consideration then tasks of verification and validation follow. The next part introduces a general mapping schema which is used for the description of the one-to-one sense relationship between the input and output range of a particular software. The test model based on this schema includes test inputs belonging to different error classes.

Verification of security-critical software systems developed in object-oriented environment

Key words: software verification, object-oriented system, iterative test generation, regression testing

This paper outlines system verification methods which facilitate the testing of large-scale and complex object-oriented systems. The presented methods were implemented in the form of a test framework and were successfully applied in testing and auditing of an application-specific operating system. These methods proved to be very efficient, the time needed for the development and realization of tests is shorter than half of the originally situation calculated without this methodology.

Digital signature with biometrics considerations

Key words: biometrics, public key cryptography, error correcting coding, channel coding

The technology of biometric digital signatures fundamentally concentrates on the strengthening of signer authentication and non-repudiation. The basic idea of our proposed method is to store the secret key encoded in a way that it can only be restored using some information extracted from the fingerprint of its owner, and only after its successful restoration can it be used for creating signatures. This way it is much harder to abuse the stolen signing device, that is, the encrypted key, since the creation of the signature requires the owner's fingerprint – besides the currently used PIN code. It is important to notice that this algorithm does not influence the usage of the public key; therefore it remains fully compatible with the recommendations of PKI, the system of certificates, the process of signature verification and so it works seamlessly with current applications.

IMAGE PROCESSING AND MEDICAL APPLICATIONS

Body swinging and tremor measuring techniques of hand

Key words: stabilometry, tremor, stress, measuring technology

The stability of bearing is maintained by a complex bio-controlling system. The controlled parameter is the position of vertical projection of center of gravity of body. In the case this point is within the basis superficies, the standing is stabile. There are several tests to prove that the system is seeking for the optimal position by which it is heading towards vertical projection of the centre of gravity, i.e. the centre of basis surface. The accuracy of regulation is highly different according to individual circumstances. When in position, it performs body swinging. By the analysis of tremor and body swinging similar or identical models can be identified and – with the exception of transducers – also the measuring procedures can be the same. In certain cases, such as in testing industrial job qualification, special adapters may be needed.

Detecting of spots on mammograms using texture analysis

Key words: medical image processing, computer aided diagnosis, texture analysis, decision tree

Today the most reliable method for detecting breast cancer is mammography. Snapshots made during mammographical check-up shall be diagnosed by two independent physicians but this can be done with computer aided methods as well. It would be very useful if the system applied would process pictures in advance, i.e. it would separate those which are surely negative and would highlight those which are suspect. The introduced algorithm forms part of the system and is used for finding tumour shadows and spots on snapshots. Within the ultimate system several – parallel – algorithms are dealing with the solution of individual problems and then the combination of these considerations form the final answer of the system.

Real-time 3D graphics in an embedded environment

Key words: 3D representation, FPGA, System-On-Chip

Typical requirements against embedded environments (low price, low energy consumption) result in the development of efficient architectures, and this is also true for the underlying hardware. This paper introduces a possible method for the improvement of the efficiency of 3D graphical representation and for the reduction of the underlying memory bandwidth.

INTELLIGENT SYSTEMS

Adaptive document management for information extraction

Key words: information extraction, document analysis

The different information and knowledge management solutions play an increasingly important role in the industry and sciences. Information extraction methods and techniques have become especially important since they support the extraction of relevant information of natural language documents. However, in order to obtain a reasonable performance, several different document management methods have to be applied in a coordinated manner. The purpose of the introduced document management system is to offer a unified theoretical and software framework for the development of applications in which the analysis and processing of natural language texts is required.

A novel method of implementation of community decisions

Key words: intelligent agencies, limited optimization, optimization of community decision

Intelligent systems play an important role in our everyday life. However, there is no comprehensive system specification principle which would allow for the coordinated planning and analysis of these systems. This problem can be approached by the unification of game, agent and evolution principles. The modeling of the decision mechanism of intelligent systems allows for a novel and more efficient implementation of agent-based communities (or rather, the implementation of community decisions).

Where we are? – Positioning in autonomous vehicles

Key words: robot navigation, particle swarm optimization

In course of past years one could meet more and more self-moving vehicles, machines and within just a few years we can use such machines even in our homes. A more thorough analysis of autonomous robots suggests that navigation is the heart of problems to be solved. These tasks can be divided into two groups: those that require collision-free movement within a particular environment and those that require expedient navigation. The bypass of obstacles remains a basic requirement.

Summaries • of the papers published in this issue