DIPLOMAMUNKA

Darlaczi Gabriella

Debrecen

2007

DEBRECENI EGYETEM

INFORMATIKAI KAR

Interaktív szolgáltatások a digitális televíziózásban – MHP alkalmazás fejlesztése

Témavezető:

Dr. Juhász István

Egyetemi adjunktus

Készítette:

Darlaczi Gabriella

Programtervező matematikus

DEBRECEN

2007

Tartalomjegyzék

1. Bevezetés

A digitális technológia már évek óta körülvesz minket, elég csak a számítógépekre, a digitális kamerákra vagy a telefonokra gondolni. A televízió világa, azonban, elég sokáig kimaradt a technológiai innovációk folyamatából, és digitális technológia helyett, sok esetben még mindig kizárólag analóg technológiát használnak. De már nem sokáig, ugyanis, ahogy sok más európai országban, Magyarországon is, 2012-ig lekapcsolják az analóg műsorsugárzást, azaz a hagyományos földi sugárzású televíziós műsorszórást, és helyébe a lényegesen több csatorna átvitelére alkalmas digitális technológia lép.

A digitális televíziózás számos előnye közül, a legnagyobb lehetőség talán az úgynevezett „interaktivitásban” rejlik. Az interaktivitás lehetőséget ad a nézőnek arra, hogy beavatkozzon a hozzá érkező adatfolyamba: kikérje a képernyőre a műsorral együtt érkező kiegészítő információkat, vagy üzeneteket küldjön (egy ún. válasz csatornán keresztül) a szolgáltató felé (pl. elektronikus vásárlások, szavazások, játékok vagy Internet-elérés során).

Ahhoz hogy digitális műsort nézhessünk, nem kell feltétlenül lecserélnünk a tévé készülékünket, elég beszereznünk egy set-top box-nak nevezett vevőkészülék (a jelenleg megvásárolható TV készülékek, melyekbe be van építve a set-top box-ok funkcionalitása meglehetősen drágák még). Az interaktivitást támogató set-top box-ok, a Java Virtual Machine platformjuk által, képesek Java alapú interaktív alkalmazások futtatására.

A dolgozat célja

A diplomamunkám témája és célja a digitális televíziózás technológiái, szabványai és lehetőségeinek bemutatása és emellett, egy set-top box-on futtatható interaktív, Java alapú alkalmazás fejlesztéséhez kapcsolódó eszközrendszerek és kérdések megtárgyalása.

A dolgozat felépítése

A második fejezet a digitális tévé innovációjáról, struktúrájáról és az általa nyújtott szolgáltatások és lehetőségekről ad áttekintést.

A dolgozatnak nem célja, hogy az egyes szabványokról és technológiákról egy minden részletre kiterjedő összefoglalást adjon, csupán egy áttekintést kínál, különös tekintettel az MHP alkalmazások elkészítéséhez felhasználható komponensekre, eszközrendszerekre kiterjedően.

A harmadik fejezetben, nagyvonalakban ismertetem a digitális televíziózás legfontosabb szabványait a Digital Video Broadcast és a Multimedia Home Platform szabványokat.

A negyedik fejezetben az MHP platformon futtatható alkalmazások állnak a középpontban. Ebben a fejezetben mutatom be az MHP alkalmazások típusait, jellemzőit, viselkedését és a fejlesztéshez használható eszközrendszereket.

Az ötödik fejezetben az MHP alkalmazások biztonsági kérdéseinek megtárgyalása következik.

A hatodik fejezetben, egy DVB-MHP interaktív szolgáltatás fejlesztéséhez kapcsolódó kérdések tárgyalása következik. A kérdések megtárgyalása konkrét példát használva történik, ugyanis szerencsém volt egy MHP-s projektben részt venni. Bemutatom az interaktív szolgáltatáson alapuló rendszer architektúráját és funkcióit, a fejlesztői környezetet és végül az MHP alkalmazások felhasználói interfészéről fogok beszélni.

2. A Digitális TV: Innováció és lehetőség

A televíziózás világában egyre nagyobb teret nyer a digitális műsorszórás, amely képes gyökeresen megváltoztatni a televíziózás szokását. A digitális televíziózás, a DTV (Digital Television) iránti hatalmas érdeklődés legfőbb oka az, hogy általa nagy mennyiségű információt el lehet juttatni, nagyon alacsony költségekkel, nagyon sok nézőhöz (felhasználóhoz).

Mit nyújt a digitális televíziózás?

- a szokásos analóg televíziós csatornában 4-6 kiváló minőségű műsor átvitele is lehetséges a szolgáltató igényétől függően.
- Kifogástalan, zajmentes képminőséget: nincs szellemkép, villódzás, színtorzulás.
- CD minőségű hangot: sztereo, Dolby Surround vagy többnyelvű kísérőhang.
- Kényelmesebb kezelhetőséget: a néző menülistából választhatja ki a nézni kívánt műsort. A kiválasztás történhet a műsor neve vagy a műsor fajtája alapján.
- A műsorcsomagban a kép- és hangjelek mellett a műsor címének, közvetítési idejének és egyéb kísérő információknak a továbbítására is lehetőség van.
- Kötetlen vételi helyeket: egyes vevőkészülékek nem igényelnek tetőantennát, csak egy rövid, kb. 10 cm-es botantennát, amelynek segítségével kiváló, szellemkép mentes lesz a vétel.
- Mobilitást: a néző ülhet akár villamoson, akár egy, az autópályán száguldó autóban, mindig tökéletes a vétel.
- Interaktivitást: visszirányú kapcsolat kialakításával - ez lehet mobil- vagy vezetékes telefon - igénybe vehetők az olyan interaktív szolgáltatások, mint például az T-government vagy az T-health.

2.1. A digitális TV struktúrája és szolgáltatásai

A digitális tévére való áttérés, mindenekelőtt, az interaktivitás megérkezését jelzi. Az interaktivitás innovációja a digitális televíziózás világában a következő profilokon keresztül valósul meg:

- **Enhanced TV**: olyan alkalmazások jellemzik, melyek a hagyományos televíziós programokat, hozzájuk kapcsolódó plusz tartalommal gazdagítják (tesztek, képek, videók), illetve az olyan alkalmazások, amelyek lehetővé teszik a nézőnek, hogy interaktívan részt vehessenek a műsorokban (közvélemény kutatások, szavazások, játékokban való részvétel);
- **Interactive TV:** előírja, hogy az alkalmazások a Válasz Csatornán keresztül (sokszor egy a dekóderbe beépített modem) kommunikáljanak a tartalomszolgáltatókkal; a felhasználó igénybevehet olyan szolgáltatásokat, mint a fizetés vagy az előjegyzés;
- **Internet on TV:** előírja az Internet elérést a Válasz Csatornán keresztül;
- **Personal TV:** lehető teszi a televíziós program személyre szabását, olyan értelemben, hogy meg lehet szabni a program választékot és a kezdési időpontokat;
- **Connected TV**: a TV összekapcsolását jelenti másik háztartási eszközökkel (számítógép, kamera, PDA); ilyen módon lehetőség nyílik a **Home Networking** megvalósítására, azaz internet szétosztására az eszközök között, vagy audió és videó fájlok küldésére, vagy akár kezelésére.

2.1.1. Kínált szolgáltatások

Egyre több országban, egyre több teret nyernek a digitális TV-s fejlesztések. A set-top box – ok által kínált lehetőségek megsokszorozódása, és mindenekfelett, a válasz csatornát megvalósító technológiák jobb kiaknázása (manapság, a legtöbb esetben, egy egyszerű modemen keresztül), valószínű, hogy a szolgáltatások, lehetőségek számának és minőségének növekedéséhez vezetnek mindazonáltal, már manapság is egyértelmű innováció van a lehetséges szolgáltatásokban:

- **Electronic Program Guide (EPG):**
  ez egy olyan magas szintű útmutató a tévénézők számára, mely lehetővé teszi a keresést, a navigációt a tévé programok között, és így a felhasználó könnyen össze tudja állítani magának kedvenc műsor listáit vagy keresni tud az egy adott órában futó műsorok között;

- **Pay per View**
  egy olyan szolgáltatás, melye lehetővé teszi, hogy a tévénézők fizetős tartalmakat is elérhessenek otthonról. A smart card olvasóval is rendelkező set-top box-ok nagyon egyszerűvé és gyorssá tehetik ezt a szolgáltatást.

- **Interaktív szociális szolgáltatások**
  ide az olyan szolgáltatások tartoznak, mint a T-commerce, T-banking, T-government, T-health; ezek mellett, lehetnek még olyan szolgáltatások, mint az időjárás, közlekedési és hasznos helyi információk közzététele.

- **Personal Video Recording (PVR)**
  ehhez a szolgáltatáshoz szükség van egy merevlemezzel felszerelt set-top box-ra. A set-top box merevlemezére felvehetünk különböző műsorokat, lementehetünk képeket, videókat, vagy letölthetünk olyan játékokat, amelyek jelenleg csak PC-re vagy játékkonzolra fejlesztett változatban érhető el;

- **Intearctive advertising**
  amellett, hogy a felhasználó valós időben tud interaktívan részt venni a műsorokban, a reklámblokkokat is személyre szabhatja. Ha valamelyik reklám érdekli, megnézhet plusz információkat a termékről vagy szolgáltatásról (hol lehet beszerezni, termékleírások), vagy ha akár rögtön meg is rendelheti (smart card-al ki is tudja rögtön fizetni).

2.2. Interaktivitás

Az interaktivitás kétségtelenül a digitális TV által nyújtott legfontosabb szolgáltatás. Számos országban, például Olaszország, Anglia, Spanyolország, nagy figyelmet szentelnek az ilyen irányú alkalmazások fejlesztésére és számos szakértő nagy jövőt lát ebben az új televíziós irányzatban. Joggal kérdezhetik azt, hogy miért használjuk a TV-t olyan szolgáltatások elérésére, amelyeket az Interneten keresztül számítógéppel is elérhetünk. A legfőbb érv, a digitális TV mellett, az hogy, a TV az az eszköz, amelyik a legtöbb háztartásban jelen van és amelyiknek kezelése okoz legkevésbé gondot az embereknek. A másik fontos tény, ami a digitális TV-s technológiák mellett áll, az hogy amint lekapcsolják az analóg műsorszórást,

szinte minden háztartásban jelen lesz a digitális TV-s sugárzás, melyek száma egyelőre jóval meghaladja az Internet eléréssel (főleg szélessávú) rendelkező háztartások számát. Tehát a TV-n keresztül jóval több emberhez juttathatunk el információkat.

2.2.1. A válasz csatorna

Ahhoz, hogy digitális műsorszórást fogjunk a tévével, egyedül egy set-top box-ra van szükségünk. Azonban ha interaktív szolgáltatásokat is igénybe szeretnénk venni, a set-top box egy válasz csatornával is fel kell legyen szerelve vagy valamilyen módon együtt kell működjön egy ilyen csatornával. A válasz csatorna (Return Channel - RC) egy olyan kétirányú csatorna, mely biztosítja a set-top box-on futó alkalmazás kommunikációját a szolgáltató központ szervereivel. A legtöbb esetben a válasz csatorna megoldása egy a dekóderben beépített egyszerű telefonos modem, azonban jobb megoldás, amely mostanság kezd elterjedni, az ADSL vagy az Ethernet kártya (már elég sok olyan set-top box van a piacon, melyekben integráltak Ethernet kártyát is). Lehetőség van GPRS-en keresztül is megoldani a válasz csatorna kérdését, mely legalább nem gyarapítja azt a rengeteg kábelt, ami már amúgy is behálózza otthonunkat.

2.2.2. Alkalmazási példák az interaktív TV szolgáltatásaira

Az interaktív TV (iTV) szolgáltatásait számos területen lehet alkalmazni (lásd az **1. Függelékben** a screenshot-okat). Az iTV alkalmazások első nagy csoportjába tartoznak a műsorokhoz kapcsolódó alkalmazások, az úgynevezett *műsorfüggő alkalmazások*. Ilyenek a kvíz műsorok, szavazások, és az egyes műsorokhoz plusz információkat szolgáltató alkalmazások is.

A második nagy csoportba a *műsor független alkalmazások* tartoznak. Ilyenek az általános információs szolgáltatásokat nyújtó alkalmazások, a kommunikációs és a szórakoztató alkalmazások, a Video-on-Demand (VoD), a T-commerce, T-government, T-learning és a T-health alkalmazások.

A következő két táblázat az interaktív szolgáltatások osztályozását szemlélteti. Az első táblázatban a szolgáltatások jellege szerinti, a második a Válasz Csatorna típusa szerinti osztályozás van.

Információs szolgáltatások	Interaktív szolgáltatások	Tranzakciós szolgáltatások
Fejlett EPG	On-demand reklám, EPG profil	T-commerce – fogyasztási termékekkel
Böngészés	Szöveges kommunikáció, communities	T-commerce – média termékekkel (pl.: zene, film)
Információs szolgálatok	Előjegyzések, információ-igénylések, operatív szolgáltatások	T-banking
Interaktív reklám	T-education, T-games	Fogadások, kaszinó

1. táblázat: Az interaktív szolgáltatások osztályozása a szolgálatások jellege szerint

Nincs Válasz Csatorna	Minimális válasz csatorna	Szélessávú válasz csatorna
Home-Shopping prepaid smart-card-al	Előadásban való részvétel, szavazások, kvízek által	Archívumok, VoD, vagy hírek böngészése
Offline quiz és egyéb játékok	SMS, chat, e-mail	Games multiplayer, videocommunities
Lokális adatbázisok (pl.: enhanced teletext, supertelevideo...)	T-government, T-banking, T-commerce (bankkártyával)	T-learning
Szoftver letöltések	Versenyek, fogadások	Személyreszabott TV, hálózati PVR

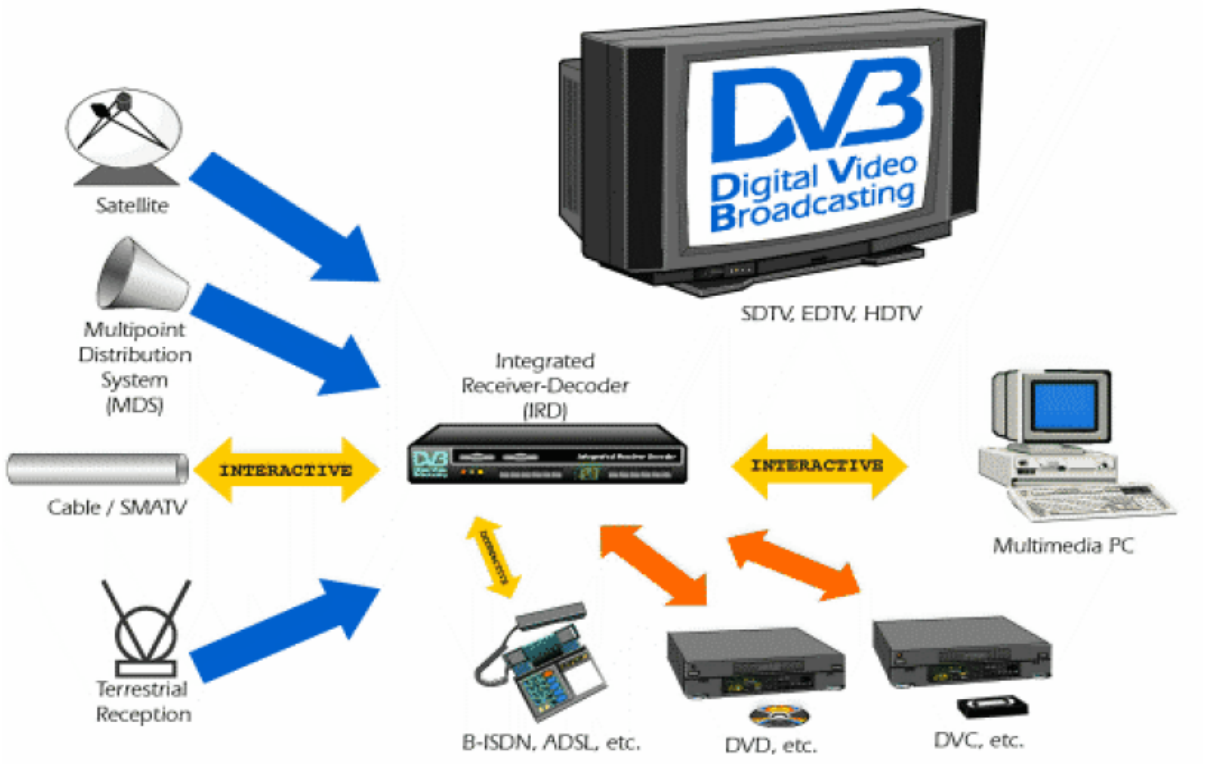
2. táblázat: Az interaktív szolgáltatások osztályozása a válasz csatorna típusa szerint

3. Az MHP és a DVB szabvány

3.1. Digital Video Broadcast

A Digital Video Broadcast Project (DVB) egy 1993-ban alakult ipari konzorcium, melynek jelenleg közel 260 tagja van: műsorszórók, gyártók, hálózatkezelők, szoftverfejlesztők, szabályozó testületek és mások közel 35 országból, akik azért álltak össze, hogy globális szabványokat hozzanak létre, a digitális műsorszórásra és a hozzá kapcsolódó szolgáltatásokra vonatkozóan.

Az első jóváhagyott szabvány, maga a DVB [4] szabvány volt: egy ipari szabványrendszer, ami az MPEG-2 szabványra épül. A DVB szabvány definiálja az összes olyan tulajdonságot, ami egy digitális műsor, videó és adatok szórására alkalmas rendszert jellemez.



1. ábra: A DVB szabvány szereplői

A DVB szabványnak alapvetően 3 fajtája létezik:

- DVB-S: DVB-Satellite, azaz a műholdas kommunikációra kifejlesztett DVB szabvány.
- DVB-T: DVB-Terrestrial, amit a földfelszíni digitális adáshoz fejlesztettek ki.
- DVB-C: DVB-Cable, ami pedig a kábeles szolgáltatók számára kifejlesztett digitális kommunikációs szabvány.

Az 1990-es évek első felében indultak be a műholdas és kábeles platformokon (DVB-S és DVB-C) a digitális szolgáltatások. A rendszeres földi (DVB-T) sugárzást 1998-ban Nagy-Britanniában kezdték el, de azóta már több európai országban is működnek kereskedelmi szolgáltatások, többek között Svédországban, Spanyolországban, Finnországban, Németországban, Hollandiában, Belgiumban és Olaszországban.

A DVB-T szabvány a VHF/UHF sávokat használja és a nagyfokú tömörítésnek köszönhetően az analóg 8 MHz sávszélességű csatornába akár 4-6 program is belefér, azaz sokkal jobb a frekvencia-kihasználás, ami nagyon nagy előny, mivel a frekvencia nemzeti kincs és a felhasználható csatornák száma véges.

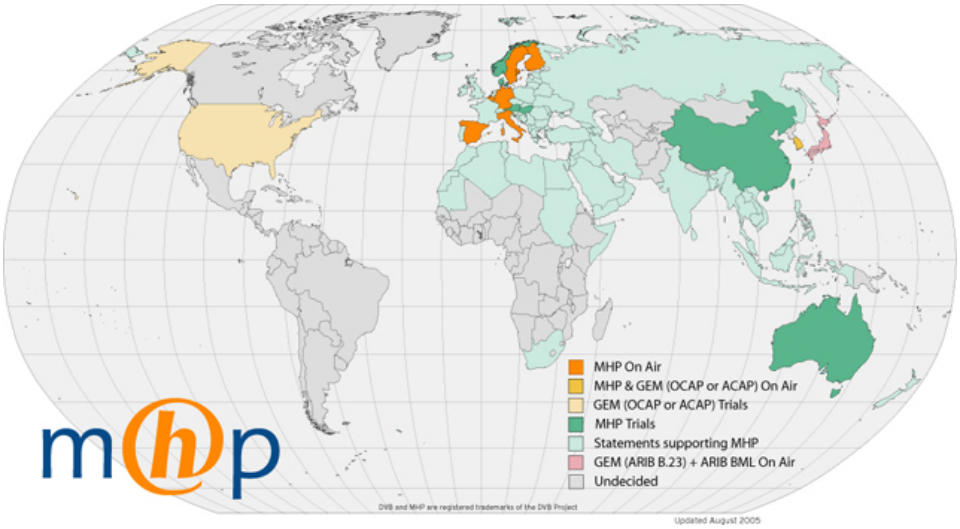
3.1.1. Set-top box

Ahhoz, hogy a digitális televíziós adásokat venni tudjuk, egy olyan készülékre van szükség, ami a vett digitális jelet átalakítja a jelenlegi analóg televíziók számára fogható analóg jellé. Ez a készülék az úgynevezett set-top box (STB). Az STB veszi az adóból érkező programcsomagot (adatfolyamot), kibontja azokat, és a televízió képernyőjén megjelenik a kiválasztott program.

Léteznek olyan televízió készülékek, melyek beépítve tartalmazzák a set-top box-ot, azaz a digitális vevődekódert- ezek az integrált digitális televíziók (IDTV - Integrated Digital Television).

3.2. Multimedia Home Platform

A Multimedia Home Platform (MHP) a DVB konzorcium által kidolgozott szabvány, a digitális televíziós platformon használható interaktív alkalmazásokra vonatkozóan.



2. ábra: Az MHP elterjedése a világban

Az egyszerűség kedvéért, MHP szabványt [5] Java API-k együtteseként definiálták. Az MHP Java API-k segítségével a TV képernyőn, MPEG-2 adatfolyamként megjeleníthető alkalmazásokat lehet készíteni. Az interaktív alkalmazásokat pedig, egy olyan STB-on lehet futtatni, amely támogatja az MHP valamely verzióját. Az MHP, definiálja tehát az alkalmazás és az alkalmazást futtató készülék, a STB, közötti interfészt. Általa az alkalmazásokat közel hardver függetlené lehet tenni. Azért csak közel hardver függetlenek lehetnek az interaktív alkalmazások, mert az MHP szabvány nem köti ki, hogy csak egyetlen implementációja lehet, és csak az szerepelhet middleware-ként az összes piacon lévő STB-on. Emiatt STB gyártónként változhat a middleware implementációja.

Az MHP-ről először egy 1994-es konferencián beszéltek, az MHP specifikáció első verzióját, azonban csak 2000-ben hagyta jóvá az ETSI [2]. Az MHP szabvány kiterjed a DVB szabvány mindhárom fajtájára: a műholdas, a kábel és a földfelszíni DVB-re is. Ezért egy DVB-MHP alkalmazás fejlesztésénél mindegy, hogy az alkalmazást DVB-T, DVB-C vagy DVB-S-t használva fogják sugározni, ugyanúgy fog működni mindhárom esetben. A DVB Project a Java-t választotta az MHP alkalmazások programozási nyelveként. Ezek után nyugodtan mondhatjuk azt, hogy platform független MHP alkalmazásokat lehet fejleszteni, pontosabban a dekóder DVB-típusától független MHP alkalmazásokat, legyen az földfelszíni,

kábel vagy műholdas digitális adás vételére alkalmas STB.

3.2.1. MHP profilok

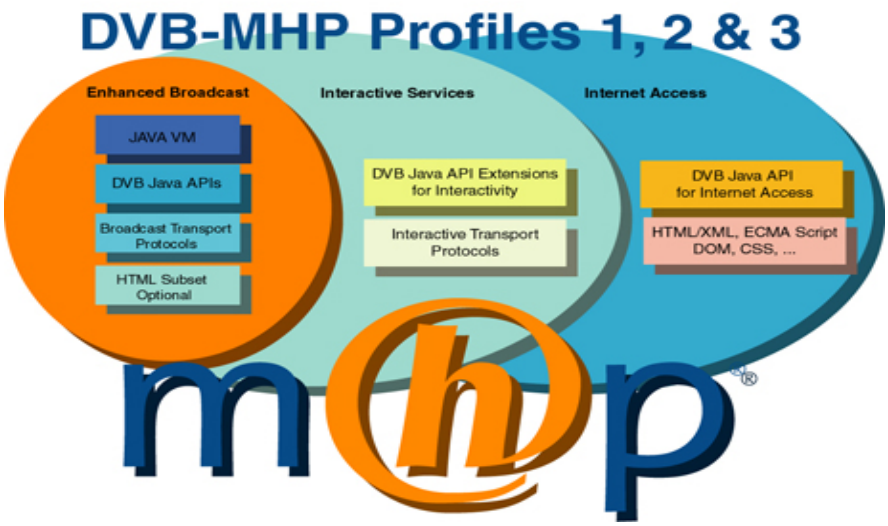
A nyílt middleware rendszer könnyebb implementációja miatt, a DVB bevezette a „Profilok” fogalmát. A profil egy alkalmazási területre vonatkozik és, ennek következményeként, a STB képességeire is. Három MHP-profil létezik, MHP specifikáció azonban csak kettő, mivel az első két profil majdnem ugyanaz, őket összevonták egy dokumentumban.

A DVB a következő három profilt definiálja:

1. **Enhanced Broadcast Profile** ES 201 812 (MHP 1.0): ez a profil engedélyt ad az audió és videó tartalmak hagyományos áramlására és arra, hogy az alkalmazások letöltődjenek a STB-ra. Ez az alap profil, és nem feltételezi a válasz csatorna jelenlétét a STB-nál;

2. **Interactive TV Profile** ES 201 812 (MHP 1.0): ez a profil feltételezi a válasz csatorna létezését (általában PSTN v.90). Az alkalmazások letölthetik az osztályokat a válasz csatornát használva, míg ez az Enhanced Broadcast Profile esetén csak a broadcast csatornán keresztül volt lehetséges;

3. **Internet Access Profile** TS 102 812 (MHP 1.1): ez a profil támogatja az olyan internetes alkalmazásokat, mint az e-mail kliensek vagy a böngészők.



3. ábra: MHP Profilok

3.2.2. MHP specifikációk

Az MHP 1.0 az alapszabvány. Itt specifikálják a digitális tévére fejlesztett alkalmazások futási környezetét, hogy az alkalmazások függetlenek lehessenek a STB hardver és szoftverétől.

Az MHP 1.0 implementálja az Enhanced Broadcast Profile-t és az Interactive Broadcast Profile-t, és a következőkből áll:

- Java Virtual Machine
- Java DVB API;
- Különböző formátumok, mint: JPEG, MPEG-2, GIF és PNG;
- broadcast protokollok;
- szállítási protokollok a válasz csatorna részére (tartalmazza az IP protokollt is)

Az MHP 1.1 szabvány az MHP 1.0 kiterjesztése, ami az Internet Access implementációjaként jött létre. A következőket tartalmazza:

- API az Internet kliens alkalmazásához;
- API a smart card-al való kommunikációhoz;
- a válasz csatornát használó letöltési lehetőség;
- DVB-HTML alkalmazások futtatása.

4. MHP alkalmazás

A technológiai megoldás szerint, az MHP alkalmazásokat két osztályba sorolhatjuk: DVB-Java alkalmazások és DVB-HTML alkalmazások. A DVB-J alkalmazásokhoz képest, a DVB-HTML alkalmazások kevésbé népszerűek, legfőbbképpen a DVB-HTML API kései megjelenése (MHP 1.1-től) és komplexitása miatt.

A DVB-J (DVB-Java) alkalmazások az MHP API-kat használva, Java nyelven készülnek, pontosabban PersonalJava-ban. A DVB-Java alkalmazások Xletekként ismertek, és ahogyan az elnevezésükből is sejthető, nagyon hasonlítanak az appletekhez. Az MHP 1.0.x a PersonalJAE specifikációra [17] épül, ami a PersonalJava Application Enviroment Version rövid változata. A Personal Java egy Java szoftverfejlesztési platform mobil és beágyazott rendszerekhez, könyvtárait és jellemzőit tekintve, pedig közel ekvivalens a Java 1.1.8.-al.

Egy másik osztályozása az MHP alkalmazásoknak a következő: *sugárzott, tárolt* vagy *rezidens* MHP-alkalmazások. A fejezet középpontjában a DVB-J alkalmazások állnak, de a sugárzott és a rezidens alkalmazások közötti kapcsolatról és az MHP 1.1-ben bevezetett tárolt alkalmazásokról is ejtek néhány szót a fejezet során.

4.1. DVB-J alkalmazások: Xletek

A hagyományos Java-alkalmazás modell nem működik túl jól egy digitális TV-s környezetben. Az alap Java-alkalmazás modell azt feltételezi, hogy egyszerre csak egy alkalmazás futhat egy adott virtuális gépen, és az alkalmazás teljes vezérléssel rendelkezik az életciklusa felett (amelyhez az is hozzátartozhat, hogy leállítja a Java VM-et, amelyiken fut). Ez cseppet sem ideális egy digitális TV jelvevő esetében, ahol egyszerre több alkalmazás is futhat, és ahol szükség van az alkalmazások egymástól való elválasztására. Szerencsére, létezik egy jó kiindulási pontunk, mely közelebb visz ahhoz, amire szükségünk van: az *appletek*. Az applet típusú programok, a hagyományos alkalmazásokkal szemben,

böngészőben futtathatóak és egyszerre akár több példányban is futhatnak ugyanazon a weboldalon.

Természetesen a digitális TV világa különbözik a Web-től, ezért ahhoz ez az elképzelés megvalósítható legyen digitális TV-n is, előbb változtatásokra volt szükség. Így jöttek létre az **Xletek.** Az appletekhez hasonló xleteket, a Sun vezette be a JavaTV specifikációban, mint standard Java-alkalmazás formát az MHP-hez és a többi digitális TV szabványhoz. Úgy, mint az appletek esetében, az Xlet interfész megengedi, hogy egy külső forrás (az application manager a digitális TV jelvevő esetében) indítson el és állítson le egy alkalmazást, valamint azt is, hogy más úton kontrollálja az alkalmazásokat. Az Xlet interfész a `javax.tv.xlet` csomagban található meg:

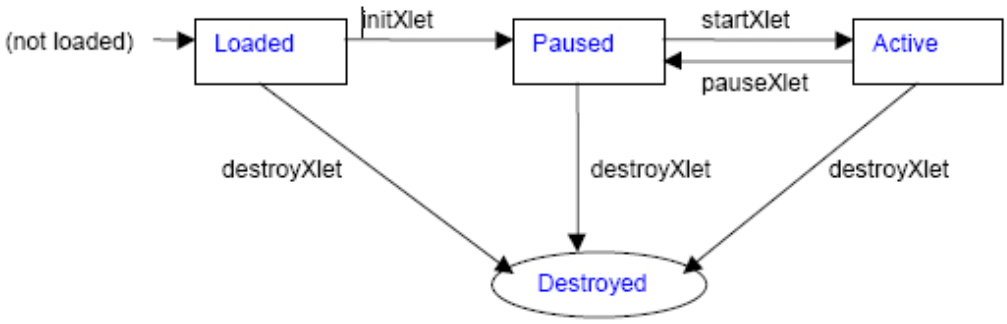
```
public interface Xlet {

  public void initXlet(XletContext ctx)
    throws XletStateChangeException;

  public void startXlet()
    throws XletStateChangeException;

  public void pauseXlet();
  public void destroyXlet(boolean unconditional)
    throws XletStateChangeException;
}
```

Ha ezt összehasonlítjuk a `java.applet.Applet` osztállyal felfedezhetünk néhány hasonlóságot. Mint az applet osztály, az Xlet osztályban is léteznek olyan metódusok, melyek inicializálják, elindítják és leállítják az xletet. Az xletek és az appletek közötti legnagyobb különbség az, hogy az xletek lehet szüneteltetni és újraindítani. Ennek oka nagyon egyszerű – egy olyan környezetben, ahol az alkalmazások egy set-top box-on futnak, a hardver megszorítások miatt, egy időben csak egy alkalmazás lehet aktív, azaz *látható*. A többi alkalmazást szüneteltetni kell, hogy legyen elegendő erőforrás az éppen a TV képernyőn *látható* alkalmazásnak.

Az xlet sokkal egyszerűbb, mint az applet – a műsorterjesztők és a set-top box gyártók paranoiája miatt, az eléggé le lettek butítva, le lettek korlátozva abból a szempontból, ami az interaktív tevékenységüket illeti.

4.1.2. Az xlet életciklusa

Az Xletnek négy fő állapotba lehet: *Betöltve*, *Szüneteltetve*, *Elindítva* és *Leállítva*. Az alábbi ábra szemlélteti az Xlet életciklusának lehetséges állapotait:



4. ábra: Az Xlet állapotdiagramja. Forrás: www.mhp-knowledgebase.org [14]

Az application manager betölti az xlet main osztályát tartalmazó fájlt, és meghívja az alapértelmezett konstruktorát, hogy példányosítsa az xletet. Ez megtörténhet bármikor miután az alkalmazás jelzése elkezdődött. Amikor ez megtörtént, az Xlet *Betöltve* állapotba kerül.

Amikor a felhasználó elindítja az xletet, azaz amikor az AIT azt jelzi, hogy az xletnek automatikusan kell elindulnia, a set-top box-hoz (STB) tartozó application manager meghívja az `initXlet()`metódust, aztán átad egy új `XletContext` objektumot az xletnek. Az Application Information Table (AIT) egy adott szolgáltatáshoz kötődő DVB leíró tábla, mely az adott szolgáltatáshoz tartozó, összes elérhető MHP alkalmazás leírására és ellenőrzésére szolgál.

Az xlet használhatja az `XletContext` példányt, hogy inicializálja magát és hogy előre letöltsön olyan nagyméretű objektumokat, mint például a képek, melyek betöltése az object carousel-ből több időbe telne. Amikor az inicializálás véget ér az xlet *Paused* állapotba kerül és készen áll arra, hogy elinduljon. Amint az `initXlet()` metódus visszatér, az

application manager meghívja a `startXlet()` metódust. Ekkor az xlet *Paused* állapotból átkerül *Started* állapotba és készen áll a felhasználóval való interakcióra.

Az Xlet futása közben, az application manager meghívhatja a `pauseXlet()` metódust. Ez visszamozdítja az alkalmazást *Started* állapotból *Paused* állapotba. Az alkalmazás újra aktívvá válik, amikor újra meghívódik a `startXlet()` metódus. Ez számtalanszor megtörténhet az xlet életciklusa folyamán.

Az xlet befejezi futását, amikor az application manager meghívja a `destroyXlet()` metódust, mely az xletet *Destroyed* állapotba helyezi, és felszabadítja az általa használt összes erőforrást. Ezután az xlet már nem indítható újra.

Fontos újra megjegyezni, hogy az xlet nem egy hagyományos Java alkalmazás, sokkal közelebb áll az appletek működéséhez. Mint az appletek esetében, egyszerre több xlet is futhat, ami azt jelenti, hogy az xletek nem vehetnek fel bizonyos viselkedéseket, olyanokat melyek globális hatással lennének a JVM-re. Például, az xletnek sohasem szabad meghívnia a `System.exit()` metódust, mert egyáltalán nem biztos, hogy csak ő az egyedüli xlet , ami a VM-en fut.

4.1.3. Xlet Contexts

Ahogyan már említettük, minden xlethez hozzá van rendelve egy környezet – a `javax.tv.xlet.XletContext` osztály egy objektuma, hasonlóan az appletekhez, ahol az `AppletContext` osztály objektuma volt az applethez hozzárendelve. Mindkét esetben, a „context” objektumok arra szolgálnak, hogy információt szolgáltassanak az alkalmazásnak a futási környezetről, illetve arra hogy, az alkalmazás státuszának változásait jelezzék a futási környezet felé.

```
public interface XletContext {
  public static final String ARGS = "javax.tv.xlet.args"
  public void notifyDestroyed();
  public void notifyPaused();
  public void resumeRequest();

  public Object getXletProperty(String key);
}
```

A `notifyDestroyed()` és a `notifyPaused()` metódusok által az xlet értesíti a dekódert, hogy be fogja fejezni vagy szüneteltetni fogja működését. Ezt az xlet arra használhatja, hogy megbizonyosodjon arról, hogy a dekóder ismeri az összes alkalmazás állapotát, és arról hogy, folytathatja működését a megfelelő feladattal. Ezeket a metódusokat, rögtön azelőtt kell meghívni, mielőtt az xlet `Paused` vagy `Destroyed` státuszba megy át, mert ellenkező esetben a dekóder úgy érzékeli, hogy az alkalmazás nincs felkészülve a státuszváltásra.

A `resumeRequest()` metódust használva, az alkalmazás kérheti, hogy `Paused` státuszból, visszakerüljön `Started` státuszba. Ez akkor történhet meg, ha egy adott esemény bekövetkezik, például ha egy bizonyos időtartam letelik, vagy ha az MPEG stream-en egy bizonyos eseményt bekövetkezik. Ez tulajdonképpen megengedi az alkalmazásnak, hogy „aludjon” egy ideig. A metódussal, azonban, az alkalmazás csak egy kérelmet juttat el a dekóder szoftveréhez (a middleware-hez), mely engedélyezi, vagy elutasítja azt, a megjelenítési és erőforrás korlátozásokat figyelembe véve.

Az `getXletProperty()` metódus lehetővé teszi az xletnek, hogy elérje azokat a tulajdonságokat, melyeket számára definiáltak a tartalomszolgáltató által sugárzott információk között. Jelenleg egyetlen tulajdonság van definiálva a JavaTV és az MHP által.

Mivel az Xlet modellben nem engedélyezett, hogy a parancssori argumentumok közvetlen az xletnek legyenek átadva, az `XletContext.ARGS` által szolgáltatott tulajdonság név megengedi az alkalmazásnak, hogy hozzáférjen minden olyan argumentumhoz, melyet az application signalling-től (az AIT) kapott. Az MHP a következő MHP-property-ket definiálja még:

`dvb.app.id` – az alkalmazás application ID-je, ahogyan az application information table-ben is szerepel

`dvb.org.id` - az alkalmazás organization ID-je, ahogyan az application information table-ben is szerepel

`dvb.caller.parameters` – az alkalmazásnak átadott paraméterek, ha az alkalmazást nem az application signalling indította el.

Az `XletContext.ARGS` és a `dvb.caller.parameters` közötti legfontosabb különbség, az hogy, az előbbi az application signalling, az utóbbi pedig, az *MHP application listing and launching API* (MHP alkalmazás listázó és elindító API) által átadott paraméterekre vonatkozik. Ha az alkalmazás nem az *MHP application listing and launching API* által lett elindítva, a `dvb.caller.parameters` egy üres string lesz.

A szokásos `System.getProperty()` metódussal le lehet kérdezni az Xlet által elérhető rendszer tulajdonságokat, melyek száma attól függően változik, hogy JavaTV, OCAP vagy MHP alkalmazásról van szó.

4.1.4. Xlet példa: „Hello world!”

Most hogy láttuk, hogyan épül fel egy xlet, írjunk is egyet! Folytatva a régi tradíciókat, a „Hello world!” típusú alkalmazást fogunk írni.

```
  /**
   * Az Xlet main osztályának implementálni kell a
   * javax.tv.xlet.Xlet interfészt, ha nem teszi meg, akkor a
   * middleware nem tudja futtatni.
   */

public class MyFirstXlet implements javax.tv.xlet.Xlet
{
 // Minden xletnek van egy xlet kontextusa, ugyanúgy, mint
 // az Appletek esetében. Ezt az MHP middleware hozza létre
 // és adja át paraméterként az xlet initXlet() metódusának

  private javax.tv.xlet.XletContext context;

  // Egy privát attribútum az aktuális állapot tárolására
  // Erre azért van szükség, mert a startXlet() metódus
  // kerül meghívásra, ha indul, és akkor is, ha újraindul
  // az Xlet.
  private boolean hasBeenStarted;

  /**
   * Minden Xletnek kell, hogy legyen egy argumentumok
   * nélküli konstruktora. Csak ezt a kontstruktort lehet
```

```
 * meghívni.
 */
public MyFirstXlet()
{
  // A konstruktor üres kell legyen.  Minden inicializálást
  // az initXlet()metódusban kell végrehajtani, vagy a
  // startXlet metódusban, ha idő-, vagy erőforrás-igényes.
  // Így a middleware könnyebben tudja kontrollálni az
  // inicializálásokat.
}

/**
 * Az xlet inicializálása.  Ez az a hely, ahol minden
 * inicializálást meg kell tenni, egyébként idő- és
 * erőforrás igényes lesz. Az XletStateChangeException
 * jelzi a futtató rendszernek, hogy nem sikerült az
 * inicializálás.
 */

public void initXlet(javax.tv.xlet.XletContext context)
  throws javax.tv.xlet.XletStateChangeException
{
  this.context = context;

  hasBeenStarted = false;

    System.out.println("The initXlet() method has been" +
    " called.  Our Xlet context is " + context);
}

/**
 * Xlet elindítása.  Most az xlet megjelenhet a képernyőn és
 * elkezdhet interaktívan működni. Mindig startXlet()-ben
 * és nem az initXlet()-ben kell elvégezni az idő és
 * erőforrás igényes feladatokat.
 *
 * Az egyik leggyakoribb buktató, az hogy a startXlet()
 * metódus vissza kell térjen a hívójához. Ez azt jelenti,
 * hogy a startXlet() fő funkcióinak egy másik szálat kell
 * indítani. Tehát, a startXlet() metódusnak egyedül annyi a
 * dolga, hogy létrehozza azt a szálat, és aztán
 * visszatérjen hívójához.
 */
```

```
  public void startXlet()
    throws javax.tv.xlet.XletStateChangeException
  {
    if (hasBeenStarted) {
      System.out.println("The startXlet() method has" +
        " been called to resume the Xlet after it's" +
        " been paused.  Hello again, world!");
    }
    else {
      System.out.println("The startXlet() method has" +
        " been called to start the Xlet for the" +
        " first time.  Hello, world!");
      hasBeenStarted = true;
    }
}

  /**
   * Az xlet szüneteltetése. Sajnos senkinek sem világos
   * (még azoknak sem akik a JavaTV specifikációját írták)
   * pontosan mit is jelent ez. Általában, azt jelenti, hogy
   * az xletnek fel kell szabadítson minden erőforrást amit
   * használ, minden szükségtelen szálat le kell állítson
   * és el kell tüntesse magát a képernyőről.
   */


public void pauseXlet()
  {
    System.out.println("The pauseXlet() method has " +
      "been called. Bedtime...");
  }


  /**
   * Xlet leállítása.  A boolean paraméter megmondja, hogy
   * az xletnek engedelmeskednie kell vagy sem a kérésnek.
   * Ha a paraméter értéke igaz, az xlet le kell álljon és a
   * metódus visszatérésével, a futtató rendszer feltételezi,
   * hogy az be is fejeződött. Ha hamis, az xlet kérheti, hogy
   * ne állítsák le, az XletStateChangeException kivétel
   * kiváltásával.
   *
   * Ha a middleware még mindig le akarja állítani az xletet,
   * újra meg kell hívja a destroyXlet()-et, igaz értékű
```

```
   * paraméterrel.
  */
  public void destroyXlet(boolean unconditional)
    throws javax.tv.xlet.XletStateChangeException
  {
    if (unconditional) {
      System.out.println("The destroyXlet() method has" +
        " been called telling the  Xlet to stop" +
        " unconditionally.  Goodbye, cruel world!");
    }
    else {
      System.out.println("The destroyXlet() method has" +
        " been called requesting that the application" +
       " stops, but giving it the choice.  So, I'll" +
        " decide not to  stop.");

      throw new XletStateChangeException(
        "Please don't kill me!");
    }
  }
}
```

4.2. Rezidens alkalmazások

A sugárzott MHP alkalmazásokkal ellentétben, a rezidens MHP alkalmazások, ahogyan arra az elnevezése is utal, a gyártók által a dekóder memóriájába beépített rezidens programok. Ennek következményeként, a rezidens programok nem kell implementálják az MHP szabványt. A böngészők, az MHP terminál konfigurációját segítő eszközök, játékok és az Xlet manager pár példa a gyártók által beépített rezidens alkalmazásokra.

Az egyes rezidens alkalmazások hozzáadása a set-top box gyártóktól függ, tehát gyártónként, esetleg típusonként változik, hogy milyen alkalmazások kerülnek beépítésre az MHP terminálokba. A sugárzott MHP alkalmazások nem képesek az ilyen típusú alkalmazásokkal való interakcióra. Indirekt kommunikáció viszont előfordulhat, amikor a sugárzott alkalmazások, olyan MHP funkciókat használnak, mint a szolgáltatás kiválasztás, mely szolgáltatáshoz az MHP egységen lévő navigátornak is van hozzáférése.

4.3. Tárolt alkalmazások

Az MHP 1.1 vezeti be a tárolt alkalmazásokat, mint egy harmadik alkalmazásfajta, a sugárzott és a rezidens MHP alkalmazások mellett. A tárolt alkalmazás, olyan sugárzott alkalmazás, melyet tárolni lehet az MHP terminálon, későbbi felhasználásra. Ez különösen akkor hasznos, ha a betöltés gyorsaságára gondolunk, nyílván sokkal gyorsabban fog betöltődni a helyi tárból, mint a data carousel-ből. Egy sugárzott szolgáltatáshoz kapcsolódó tárolt alkalmazás, annak ellenére, hogy le van tárolva az MHP terminálon, csak akkor indulhat el, ha az az Application Information Table-ben sugárzott szolgáltatásként jelzik. Ugyanez vonatkozik a független alkalmazásokra is (pl.: játékok), mivel ők egy tárolt szolgáltatáshoz kötődnek, melynek ugyanúgy van AIT-je.

A tárolt alkalmazások életciklusa és viselkedése, azonkívül hogy gyorsabban töltődnek be, hasonló a sugárzott alkalmazásokéhoz.

4.4. Grafikai modell

A grafikai modell talán a legkomplexebb része az MHP specifikációnak. Ennek oka az, hogy a televíziós grafika sok ponton különbözik a számítógép grafikától [1]. Ezekből néhány dolog, amit általában figyelembe kell venni:

- *pixel arány (pixel aspect ratio)*: a számítógépes grafikával ellentétben, a video és TV alkalmazások általában nem négyzet alakú pixelekkel dolgoznak.
- *átlátszóság és áttetszőség*: hogyan tudjuk átlátszóvá varázsolni a grafikát, azért hogy a néző láthassa mi van alatta?
- *színek*: hogyan tudjuk a Java által használt RGB színskálát megfeleltetni a TV által használt YUV színskálának?
- *nincs window manager*: a window managerek túl komplexek ahhoz, hogy egy dekóderen használhatóak legyenek, ezért az alkalmazások máshogyan jutnak hozzá a területhez, amire kirajzolódhatnak.

- *user interfész különbségek*: a felhasználónak csak egy távirányító áll rendelkezésre, sokkal korlátozottabban mozoghat, mint egy PC-s alkalmazásban.

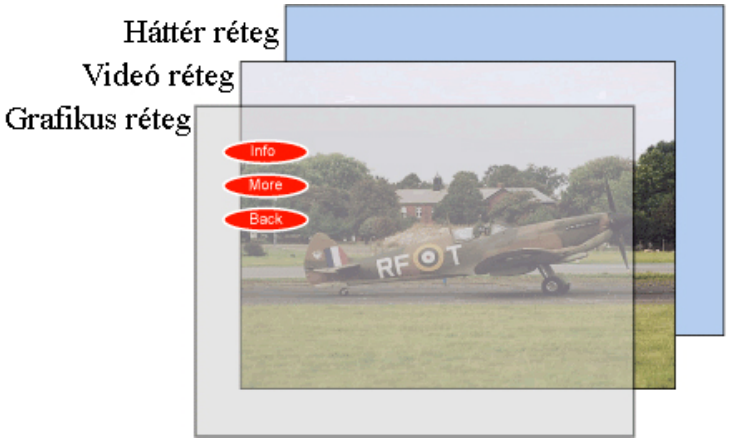
A fentiekből kiindulva, egyáltalán nem meglepő, hogy fejlesztőként, van egynéhány dolog, amit majd másképpen kell megoldani. Azonban, a legtöbbször igaznak bizonyult az, hogy ha valaki tud AWT alkalmazásokat fejleszteni, nem sok gondja lesz az MHP grafikájához kapcsolódó API-k használatával sem.

Az MHP a personalJava AWT osztályok egy olyan részhalmazát használja, melyben nem szerepelnek az ablakos rendszertől (windowing system) függő osztályok. A két grafikai modell közötti legtöbb különbség, az MHP kiterjesztéseiből adódik. A HAVi (Home Audio Video interoperability) [6] szabvány definiálja a *HAVi Level 2 GUI* –ként ismert kiterjesztést, mely tartalmazza azt az új eszközrendszert, amivel kiküszöbölhető az ablakos rendszer szükségessége. Ezen kiterjesztések az `org.havi.ui` csomagban találhatók meg, melynek néhány elemét a későbbiekben meg is fogunk nézni, előtte ám, meg kell értenünk az MHP dekóder grafikai modelljét.

4.4.1. Az MHP megjelenítési modellje

Az interaktív TV képernyője lényegében három rétegre osztható fel. A *háttér réteg*, általában egy szín, szerencsésebb esetben egy kép megjelenítésére képes. A háttér réteg fölött található a *videó réteg*, mely a videók megjelenítésére alkalmas réteg. A két réteg tetején, pedig a grafikus *réteg található*.

A grafikus réteg az, amelyiket az MHP-beli grafikus műveletek meg fognak rajzolni. A grafikus réteg felbontása különbözhet a videó vagy a háttér rétegek felbontásától, sőt akár a pixelek formája is eltérhet (a videó réteg pixelei általában szögletesek, a grafikai réteg pixelei, pedig négyzet alakúak). A set-top box természetéből adódóan, manapság még nem várhatunk el csúcsminőséget a megjelenítésben: a legjobb esetben is a grafikus réteg 256 szín megjelenítésére képes, a felbontás, pedig 320x200-as körül mozog.



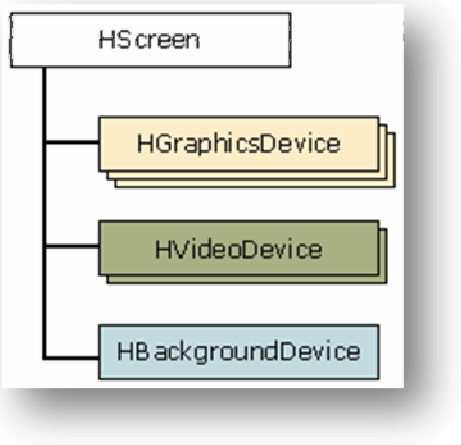
5. ábra: Az MHP grafikus alrendszerének rétegei.

Mindhárom réteget külön-külön lehet konfigurálni, ráadásul annak ellenére, hogy különböző komponenseket jelentenek a rétegek generálásáért felelős rendszerben, a rétegek konfigurálása nem független egymástól: egyik réteg valamilyen konfigurációja, megszorításokat eredményezhet másik két réteg konfigurációjára. Ezt a problémát oldja meg a HAVi (és az MHP) a `Hscreen` osztály definiálásával. A `Hscreen` egy fizikai képernyő eszközt reprezentál, és minden MHP jelvevőnek annyi `Hscreen` fizikai képernyő eszköze lesz, ahány (TV) képernyő van hozzá csatlakoztatva – azonban, általában egy képernyő lesz.

Egy `Hscreen` objektumhoz több `HScreenDevice` objektum tartozik, melyek a különböző rétegek reprezentációjára szolgálnak. Általában, minden `Hscreen`-hez a következő három `HScreenDevice` alosztály tartozik:

- `HBackgoundDevice`, mely a háttér réteg reprezentációját szolgálja;
- `HVideoDevice`, a videó réteg reprezentációjára;
- `HGraphicsDevice`, a grafikus réteg reprezentációjára.

Az MHP megjelenítési modellje egyetlen háttér eszközt enged meg, azonban videó és grafikus eszközből több is lehet, ahogyan az alábbi ábra is mutatja:



5. Ábra: Az MHP képernyőt alkotó eszközök

4.4.2. A magas szintű grafikát érintő kérdések

Az MHP a következő kép formátumokat támogatja: GIF, JPEG, PNG, MPEG I-frame. Ezenfelül kezeli a DVB – feliratot és a feliratokra vonatkozó DVB – teletext formátumokat. Ahogyan azt sejthettük, a képeket sem lehet kezelni a `java.awt.Image` objektumokkal. A feliratok manipulálására az `org.davic.media.SubtitlingControl` osztályt kell használni. Ez egy Java Media Framework (JMF) vezérlő, mely lehetővé teszi, hogy a felhasználó be vagy ki kapcsolja a feliratozást, lekérje a feliratozás státuszát, vagy kiválassza a nyelvet.

4.4.3. Az MHP grafikus eszközrendszere

Mint már említettük, az MHP drasztikusan leszűkítette az AWT csomag tartalmát, ám a legtöbb, az AWT csomagból kihagyott osztálynak, van MHP-beli megfelelője. Az MHP verziók csak a „könnyűsúlyú” komponensek használatát engedélyezi, olyanokét mint a `java.awt.Button`, `java.awt.Dialog` és a `java.awt.Menu`.

Az MHP részeként tekinthető HAVi Level 2 GUI szolgáltatja azt az eszközrendszert, melyet az AWT csomagból kihagyott osztályok helyettesítésére lehet használni. Azok a standard elemek, melyek jelenléte elvárás a GUI rendszerekben, mind megtalálhatóak a HAVi API eszközrendszerében is: *gombok, check-box-ok, rádió gombok, ikonok, görgethető listák, dialóg dobozok, szövegmezők*. Természetesen ez a felsorolás nem teljes, a teljes eszköz

készletről bővebben az `org.havi.ui` csomagban tájékozódhatunk.

Az alkalmazás felhasználói felületének tervezésével foglalkozók egyik problémája a standard eszközrendszerekkel, hogy rákényszerítenek egy standard look&feel-re (kinézetre), ráadásul a TV-s környezet technológiai megoldásai miatt, az MHP-alkalmazások esetében még több korlátba ütközünk.

HAVi API eszközrendszere lehetővé teszi, hogy az MHP-alkalmazások look&feel-jét könnyen megváltoztassuk. Az `org.havi.ui.Hlook` interfész és alosztályai megengedik az alkalmazás bizonyos GUI osztályainak, hogy felüldefiniálják a `paint()` metódusát, így nincs szükség újabb GUI osztályok definiálására az új look&feel létrehozásához. A `Hlook` objektumokat csak az `org.havi.ui.Hvisible` osztállyal vagy alosztályaival lehet használni, de így is elég sok lehetőségünk marad.

A `HLook.showLook()` metódust, a `Hvisible` osztály vagy valamely leszármazott osztályának `paint()` metódusa hívja meg, tehát nem szükséges külön meghívni. A `HVisible.setLook()`metódus egy `Hlook` objektumot vár paraméterként, és beállítja a `HVisible`-t, úgy hogy, a `Hlook` objektumot használja a `paint()` metódusa helyett.

Az olyan alkalmazások esetében, ahol a felhasználói felület elég komplex, mint például a T-commerce alkalmazások vagy email kliensek esetében, a navigáció elég nagy gond, mivel általában erre csak egy távirányító áll rendelkezésre. Ebből kifolyólag a designereknek nagyon oda kell figyelniük, milyen felhasználói felületet terveznek, mert ami PC-n jól mutat egyáltalán nem biztos, hogy televízión is jól fog mutatni, már ha az a design egyáltalán megvalósítható televízión is.

4.5. Input, event, focus

A digitális TV-re készült alkalmazások vezérlésére, általában csak egy távirányító áll rendelkezésünkre. A távirányítóval kell megoldani a navigációt és az alfanumerikus szöveg bevitelét is. A kettő közül a szöveg bevitel nyilvánvalóan körülményesebb a távkapcsoló korlátai miatt, a navigáció egyértelműen a távkapcsoló fel, le, jobbra és balra gombjaival történik.

4.5.1. Megoldások az alfanumerikus szöveg bevitelére

Több módon is meg lehet oldani az alfanumerikus szövegek bevitelét, a legjobb, ha az alkalmazásban több módszert implementálunk, hogy a felhasználónak legyen lehetősége kiválasztani, melyik a neki kényelmesebb megoldás (az Ugo Bordoni Alapítvány ajánlásai a felhasználói felület elkészítéséhez [9]).

Az első megoldás szerint, a távirányító billentyűzetét úgy használhatnánk, mint a telefon billentyűzetét, mintha SMS-t írnánk. Ez a megoldás azonban, elég körülményes főleg az idősebb korosztálynak. Lehetőség van a T9 egyszerűsített szövegbevitel használatára is, de így sem lesz ez a legjobb megoldás.

A második megoldás az úgynevezett „slot machine” módszer. Ebben az esetben a karakterek bevitele úgy történik, hogy az egyes karaktereket egy listából választjuk ki, úgy hogy görgetjük a listák a távkapcsoló fel ↑ és le ↓ nyilaival, amíg meg nem találjuk a kívánt karaktert. Ha tovább akarunk menni, megnyomjuk a jobbra → billentyűt, ha törölni akarunk, akkor a balra ← nyilat használjuk.

A harmadik megoldás egy virtuális billentyűzet használatát javasolja. Az engedélyezett karakterekből összeállítunk egy virtuális billentyűzetet, melyben a távkapcsoló jobbra, balra, fel, le billentyűkel lehet navigálni, az OK gombbal, pedig kiválasztani a megfelelő karaktert.

4.5.2. Események és input fókusz

A felhasználói által bevitt adatokat az MHP dekóder a távirányítótól kapja meg. Az MHP szabvány definiálja azt a minimális billentyű készletet, mely minden MHP dekóder távirányítóján kötelező, hogy szerepeljen. Az MHP dekóder a következő módon definiálja a távirányító egyes gombjai által kiváltott eseményeket:

Input event	MHP terminál billentyűi
VK_0, VK_1 …, VK_9	A tíz számozott gomb
VK_UP	4 nyíl billentyű + 1 kiválasztó gomb
VK_DOWN	
VK_LEFT	
VK_RIGHT	
VK_ENTER	
VK_TELETEXT	Teletext gomb
VK_COLORED_KEY_0	4 színes gomb
VK_COLORED_KEY_1	
VK_COLORED_KEY_2	
VK_COLORED_KEY_3	

3. táblázat: A távirányítóval kiváltható események táblázata

A definiált billentyű események teljes listáját, az MHP specifikáció `org.havi.ui.event.HrcEvent` osztályában találhatjuk meg.

Az AWT esemény modellébenben a billentyűzet eseményeket mindig az a komponens fogadja, amely a beviteli fókusszal rendelkezik. Ez a megszorítás azonban az MHP alkalmazásokban problémát okozhat. Képzeljünk csak el az EPG esetét, amikor egy gomb megnyomására felugrik egy pop-up ablak. Ekkor, ahhoz hogy az alkalmazás egyik láthatatlan komponense figyelni tudja az AWT eseményeket, ennek a láthatatlan komponensnek egy `Hscene`-ben kell lennie. Ahogyan azt már tárgyaltuk az MHP grafikus modelljéről szóló részben, a `Hscene`-ek nem fedhetik át egymást a képernyőn. Ez azt jelenti, hogy feleslegesen foglalná le az erőforrásokat a többi alkalmazás elől.

A probléma megoldására hozták létre az `org.dvb.event` csomagot. A csomag olyan eszközöket tartalmaz, melyek segítségével a billentyűzet események fogadására nincs szükség beviteli fókuszra. `A UserEventListener` interfészt implementáló osztályok objektumai fogadhatják az eseményeket anélkül, hogy beviteli fókusszal rendelkeznének.

Az alkalmazást úgy készíthetjük fel billentyűzet események fogadására, hogy definiáljuk a `UserEventRepository` osztályban azokat az eseményeket, melyeket az alkalmazás, futása során, fogadni kíván. A definiált események eléréséhez, az alkalmazás az `EventManager` osztályt használhatja. Az `EventManager.getInstance()`

metódussal megkapjuk az osztály egy példányát, aztán az `addEventListener()` és a `removeUserEventListener()`metódusokkal hozzáadhatunk vagy eltávolíthatunk eseményfigyelőket a `UserEventRepository()` objektumhoz.

5. Az MHP biztonsági kérdései

A digitális televízióadás sugárzói és a set-top box gyártók nagy hangsúlyt fektetnek az MHP technológia biztonsági kérdéseire. A két legfontosabb cél közül az egyik, hogy megakadályozzák az illetéktelen hozzáférést a fizetős szolgáltatásokhoz, a másik pedig, az hogy a terjesztés előtt, senki se tudja befolyásolni, meghamísítani a tartalmat.

Mint ahogy az általában szokott lenni, a biztonság nem csak a szolgáltatók érdeke, hanem a felhasználóké is. Gondoljunk csak az egészségügyi rendszerekre, ahol a betegadatok kezelésére komoly jogszabályok léteznek, vagy a T-commerce alkalmazásokra, ahol pénzügyi tranzakciók történnek.

A fizetős tartalomhoz való illetéktelen hozzáférés megakadályozása nem egy új keletű probléma a televízió világában. Az analóg és a digitális tartalomszolgáltatók egyaránt, alkalmaznak már egy ideje, olyan védelmi technikákat, mint a kódolás vagy a sugárzás zavarása. Ezek a technikák nem sokat fognak változni az MHP platformokon, nem úgy, mint az interaktív MHP alkalmazások, ahol szükség volt MHP specifikus megoldásokat is bevezetni.

A fogyasztóknak szánt alkalmazásoknak olyan megbízhatóknak kell lenniük, amennyire csak tudnak, mert a PC-n megszokott problémákat, a felhasználók jóval nehezebben fogják tolerálni, főleg az idősebb korosztály, ugyanis nincsenek megszokva azzal, hogy televízió nem működik megfelelően. Ebből az okból, az MHP dekóderekbe számos biztonsági mechanizmust építettek be.

5.1. Az alkalmazások authentikációja

A dekóderekbe beépített mechanizmusok közül, az egyik az alkalmazások authentikációs mechanizmusa. A tartalomsugárzó digitálisan alá tudja írni az alkalmazást, és ilyen módon a dekóder megbizonyosodhat arról, hogy az alkalmazást nem hamísította, vagy módosította senki és a megfelelő formában töltődik le. Az aláírás folyamata három

mechanizmusból áll:

- Hash fájlok, melyek egy ellenőrzőösszeget szolgáltatnak a fájloknak és a könyvtáraknak.
- Aszimmetrikus algoritmussal kreált digitális aláírások.
- Publikus kulcs tanúsítványok.

5.1.1. Hash állományok

A hash fájlok egy ellenőrzőösszeget biztosítanak a fájlok vagy a könyvtárak számára, ami azért hasznos, mert ki lehet szűrni, hogy sérült-e a tartalmuk. Az alkalmazott hash fájl neve `dvb.hashfile`, és minden könyvtárhoz egy hash fájl tartozik, mely az adott könyvtárban található fájlokhoz asszociált hash értékekből épül fel. A hash érétkek kiszámolására az MD5 és az SH-1 algoritmusok egyikét lehet használni.

5.1.2. Aláírás állomány

Az alkalmazás aláírása során, tulajdonképpen az alkalmazás fő hash-állományának aláírásáról van szó. A fő hash-állomány, mely az alkalmazást tartalmazó könyvtárban található, az alkalmazás könyvtáraiban és alkönyvtáraiban lévő hash állományokból összeállított hash fájl.

A tartalomsugárzók, miután kiszámolták a fő hash állomány hash értékét, a kapott értéket egy publikus kulcsú algoritmussal lekódolják. A sugárzók privát kulcsukkal lekódolják az adatokat, a dekódoláshoz használható publikus kulcsot az alkalmazás könyvtárstruktúrájának egy állományában a digitális aláírást, pedig a `dvb.signature.<id_number>` nevű fájlba helyezik el. Az `<id_number>` által (`ami` egy decimális egész szám), a különböző hatóságok alá tudják írni az alkalmazás könyvtárstruktúráját.

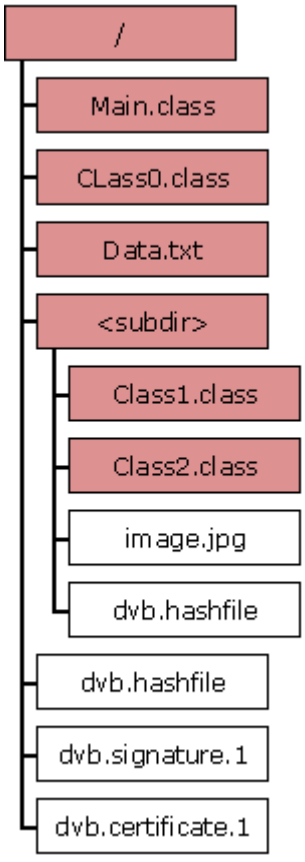
Ahhoz, hogy ez a módszer működjön, a dekódereknek meg kell tudniuk különböztetni az aláírt alkalmazásokat a nem aláírt alkalmazásoktól. Szerencsére ezt az alkalmazás ID-k alapján egyszerűen meg lehet mondani: ha az alkalmazás ID a 0x000 és 0x3FFF

intervallumba tartozik, akkor az alkalmazást nincs aláírva; ha a 0x4000 és 0x7FFF intervallumba van, akkor az alkalmazást digitálisan aláírottnak tekinthetjük.

5.1.3. Tanúsítvány állomány

Ahogyan azt már említettük, a tartalomterjesztőnek a dekódoláshoz szükséges publikus kulcsot bele el kell helyeznie a könyvtárstruktúrában. Ez a publikus kulcs a `dvb.certificate.<id_number>` „tanúsítvány” állományban lesz elhelyezve, ahol az *id_number* ugyanaz, mint a hozzátartozó aláírás állományban.

A továbbiakban megvizsgáljuk, hogyan zajlik le az aláírás folyamata. Legyen a következő könyvtárstruktúra:



8. ábra: Egy MHP könyvtárstruktúra biztonsághoz kapcsolódó állományai

Forrás: http://www.tvwithoutborders.com/tutorials/mhp [8]

Piros színnel jelöltük a fájlokat, melyeket alá akarunk írni. E könyvtárstruktúra aláírása során a következők történnek:

Minden egyes hatóságnak, mely alá kívánja írni a könyvtárstruktúrát, lesz egy tanúsítványa, melyet a root könyvtárba helyeznek el. A fenti könyvtárstruktúra egyetlen aláíró hatósága mi leszünk, ezért egy tanúsítvány lesz. Ez a `dvb.certificate.1` állományba lesz letárolva.

Az ellenőrzőösszeg készítés az alacsony szintű könyvtáraknál kezdődik. A mi esetünkben a `subdir` könyvtár két class fájlját authetikáljuk, az eredmény, pedig a `dvb.hashfile` állomány lesz, melyet a `subdir` könyvtárba helyezünk el.

A magasabb szintű hash állományok mindig az adott könyvtár alkönyvtárainak hash-állományaiból készül. Következő lépésként, elkészítjük a `subdir` szülőkönyvtárának hash állományát, a `subdir` hash-állományát felhasználva.

Mindig a könyvtárstruktúra tetején elhelyezkedő hash-állományt fogjuk aláírni. Tehát a legfelsőbb szintű `dvb.hashfile` állomány hash értékét RSA algoritmussal lekódoljuk és elhelyezzük a `dvb.signature.1` fájlba. Az RSA algoritmus publikus kulcsa a `dvb.certificate.1` állományban lesz.

Fontos megjegyezni, hogy a tartalomszolgáltatóknak oda kell figyelniük az alkalmazások aláírásánál. Mivel a set-top box-ok processzor teljesítménye még sok kívánnivalót hagy maga után, ha nem hatékonyan oldják meg az aláírás folyamatát, eléggé lelassulhat az alkalmazás betöltése. A lassulást az okozhatja, ha az ellenőrzés során túl sok fájlt kell egyszerre betölteni, és ezt azért, mert az alkalmazás állományszámához képest nagyon kevés a hash-állomány.

5.2. Az alkalmazás biztonsági modellje

Pár megszorítást leszámítva, az MHP alkalmazásoknak biztonsági modellje alapjában véve ugyanaz, mint a PersonalJava-ban specifikált biztonsági modell. Az MHP API-k számos kivételt definiálnak, melyeket akkor lehet kiváltani, ha az adott alkalmazásnak nincs elegendő joga valamely művelet végrehajtására.

Az aláíratlan alkalmazások csak az alap jogokkal fognak rendelkezni. Az aláírott alkalmazások is eleinte csak az alap jogokat kaphatják meg, de a tartalomsugárzók vagy a fejlesztők alkalmassá tehetik az alkalmazást arra, hogy kérhessen plusz jogokat a rendszertől.

A jogosultságokat érintő kérések egy XML állományba vannak összeszedve, az MHP specifikációban megadott formátumban (egy az MHP specifikációból kiragadott példa az 1. Függelékben található meg). Az xml fájl neve `dvb.<initial_file_name>.perm`, ahol az `<initial_file_name>` DVB-J alkalmazás esetén az alkalmazás fő állományának a neve, DVB-HTML alkalmazás esetén, pedig első HTML állomány neve.

Attól, hogy az alkalmazások kérhetnek bizonyos magasabb szintű jogosultságokat, még nem biztos, hogy meg is kapják, mert a jogosultságok kezelése dekódertől is függ. Amit az egyik dekóder alapból felhasználói preferenciának ítél meg, lehetséges, hogy arra egy másik dekóder minden alkalommal rá fog kérdezni. Tehát, ahhoz hogy egy jól működő alkalmazást készítsünk, a fejlesztésnél ezeket a dolgokat is figyelembe kell venni.

5.3. Feltételes hozzáférés

Főleg a fizetős tartalmak levédése végett, sok szolgáltató által használt technika a sugárzott jelek zavarása. A tartalom hozzáféréséhez, a zavarást feloldani képes eszközrendszerrel kell rendelkeznie a dekódernek. Ez az eszközrendszer, a *conditional access system (CA)*, azaz a feltételes hozzáférést kezelő rendszer.

Az MHP szolgáltatásokhoz általában feltételes lesz a hozzáférés, tehát szükség lesz az ilyen esetekben használatos eszközrendszerre, a CA API-ra.

Az MHP szempontjából a CA két részből áll:

- *CA hardver modulok* a zavarás feloldására a stream-en. Ezek magukat a modulokat reprezentáló, menedzselő osztályok, melyek a CA hardver modulok erőforrás menedzsmentjével és a jelek dekódolásával foglalkoznak.
- *ember-gép interfész*, mely a CA-ra vonatkozó kérdések végett, a felhasználóval való kommunikációt szolgálja. Ez a TV képernyőn megjelenő dialógusokból és üzenetekből áll.

Ezekről részletesebben a következő fejezetekben fogunk beszélni, előbb viszont lássuk nagyvonalakban, hogyan is működik a CA-rendszer.

5.3.1. A CA rendszer működése

A digitális műsorszórás világában, a zavarás technikája kétféleképpen is alkalmazhatjuk: zavarhatjuk a teljes adatfolyamot vagy az adatfolyam különálló elemeit. Ha a második megoldást választjuk, akkor a csomagok fejlécét épségben kell hagyni, hogy a dekóder megfelelően működhessen.

A kódolás érdekében, a CA rendszer kéttípusú adatot (*CA-üzeneteknek* is szokták nevezni őket) csatol még az eredeti adatfolyamhoz: egyik az *Entitlement Control Messages (ECM)*, a másik az *Entitlement Management Messages (EMM)*. A két CA-üzenet típus által a felhasználók képesek lesznek zavart tartalmakat nézni.

A zavarás/zavarás feloldás folyamat három információra épül: *control word*, *service key* és a *user key*. A service key lehet közös a felhasználók körében és általában egy kódolt szolgáltatáshoz egy service key tartozik. A control word egy ECM-ben kerül elküldésre - körülbelül két másodpercenként küldik, és ő adja a szolgáltatás dekódolásához szükséges információt a dekódernek. A user key segítségével lehet megállapítani a felhasználó jogosultságát a szolgáltatásra. A felhasználóként egyedi user key-el lekódolják a service key-t, és egy EMM részeként küldik el (körülbelül 10 másodpercenként küldik újra).

Amikor a dekóder egy CA-üzenetet kap, átadja a CA-rendszernek. Egy EMM esetén, a dekóder megnézi, hogy valóban neki szántak-e az üzenetet, általában úgy, hogy megvizsgálja a CA-sorszámát vagy a smart card számát. Ha neki szánták, a user key másolatát fogja használni a service key dekódolásához.

A service key-t aztán arra használják, hogy a fogadott ECM-ket dekódolják, és hogy a control word-ot visszafejtsék. Amint megvan a control word, a dekóder inicializálhatja CA-hardvert a zavart tartalom visszaállítására.

Ahhoz, hogy a z EMM-ek helyesen legyenek generálva, a CA-rendszernek tudnia kell melyik tartalomhoz melyik előfizető férhet hozzá. Erre használják a Subscriber Management System-et (SMS). Az SMS egy a felhasználók és jogosultságaik nyilvántartására használt nagyméretű adatbázis, mely hozzá van kapcsolva a CA-rendszerhez és valamilyen számlázó rendszerhez is.

Egy DVB dekóder több CA-modult is tartalmazhat, melyek logikailag ugyan azonosak, de előfordulhat, hogy különböző CA-rendszerek kezelésére is alkalmasak. A

különböző CA-modulok kezelésére, a DVB definiálta a DVB Common Interface-t (CI). Általa a dekóder biztonságosan váltogathatja a CA modulokat.

5.3.2. A dekódoló rendszer hardver interfésze

A zavart tartalom feloldásához, szükségünk van néhány speciális hardver eszközre és egy speciális dekódoló algoritmusra is. A legtöbb set-top box esetén, a CA-rendszerek a tartalom dekódolására, főleg a költséghatékonysága és egyszerűsége miatt, egy *smart card*-nak nevezett kártyát (és hozzá egy olvasót) használnak. A smart card-al való kommunikációért és a CA-üzenetek kezeléséért, a middleware részeként implementált szoftver a felelős.

Önmagában a smart card, azonban nem jelent teljesen jó megoldást, ugyanis, a smart card egyetlen CA-rendszerhez köti a felhasználót. Ez főleg akkor jelent problémát, ha a felhasználó olyan helyre viszi a dekóderét, ahol a tartalomszolgáltató másik CA-rendszert használ. Ennek megoldásaként, születtek a DVB és a CableLabs által közösen kifejlesztett szabványok a csatlakoztatható CA-modulokra: a CableCARD a CableLabs esetében és a DVB Common Interface (DVB-CI) a DVB esetében. Mindkét megoldás ugyanolyan formátumú, a STB-hoz csatlakoztatható modult használ, egy PCMCIA-kártyát. Ez a megoldás ugyan lehetővé teszi az átjárhatóságot a különböző CA-modult használó szolgáltatók között, de plusz költségeket vet fel.

A CA-API szempontjából a két megközelítés egyenlő, mert minden CA-modul egyszerre csak egy szolgáltatás dekódolására alkalmas. Minden fizikai modult a `CAModule` osztály egy példánya reprezentál.

Ha egy alkalmazásnak dekódolnia kell egy adatfolyamot, a `DescramblerProxy` osztálytól kérhet hozzáférést az ehhez szükséges erőforrásokhoz. Ez egy erőforrás proxy, tehát egy alkalmazás több példányt is létrehozhat belőle - nincs egy az egyhez megfeleltetés a `DescramblerProxy` példányok és a fizikai CA modulok között.

Miután az alkalmazás létrehozta a `DescramblerProxy` objektumot, a dekódolás elindítására a `startDescrambling()` metódust lehet használni. A dekódolás elindítása után történik az erőforrások lefoglalása is, melyek kezeléséért a `CAModuleManager` osztálya felelős.

Létezik számos esemény, melyeket a CA-rendszer generál azért, hogy információt szolgáltasson a zavarás feloldás folyamatáról és a hardver állapotáról. A `DescramblerProxy` osztály által, az alkalmazás regisztrálhat egy vagy több `DescramblerListener` objektumot, melyek az adott proxy által végrehajtható dekódolásnak specifikus üzeneteket fognak fogadni. Ehhez hasonlóan, az alkalmazás a `CAModuleManager` osztály által, egy vagy több `CAListener` objektumot is tud regisztrálni. Ezekhez, a CA-hoz kapcsolódó, általánosabb események fognak befutni, olyanok, például, mint a CA-modulok elérhetősége vagy olyanok, melyek a CA-rendszer és a felhasználó közötti interakciókra vonatkoznak.

5.3.3. Az ember és gép közötti interfész

A *man-machine interface (MMI)* az ember és gép kommunikációját szolgáló interfész. Az MMI olyankor jut szóhoz, amikor egy adatfolyam dekódolásához, a CA-rendszernek kommunikálnia kell a felhasználóval. Ez akkor fordulhat elő, például ha a felhasználó vásárolni akar egy szolgáltatást és PIN-kódot vagy valami hasonló aktivációs kódot kérnek tőle.

Az általában dialógus alapú kommunikációra, a CA API nem ad pontos definíciót, de sok lehetőséget biztosít az MMI megfelelő megjelenítésére. Ha egy alkalmazás implementálja az `MMIListener` interfészt és `MMIListener`-ként van regisztrálva a `CAModuleManager`-nél, a CA-rendszertől olyan események fognak hozzá befutni, melyek a felhasználóval történő MMI interakciókat írják le. Ezek az események az `MMIEvent` osztály példányai lesznek.

A `StartMMIEvent` értesíti az alkalmazást, ha a CA-rendszer kommunikálni akar a felhasználóval. Míg a kommunikáció *look and feel*-jéről az alkalmazás gondoskodik, az interakció típusát már a CA-rendszer mondja meg. A `StartMMIEvent` osztály, az `getMMIObject()` metódussal lekérdezett `MMIObject` attribútuma, mondja meg milyen típusú dialógust kíván szolgáltatni a CA-rendszer. A típusok a következők lehetnek: `Menu`, `Text`, `Enquiry` és `List`. Ezeket a típusokat reprezentáló osztályok számos mezővel rendelkeznek, melyeket a CA-rendszer arra használ, hogy adatokat küldjön az alkalmazásnak.

Példaként nézzük meg az Enquiry osztályt! Ez egy olyan dialógust reprezentál, ahol a felhasználótól válasz információkat várunk egy kérdésre:

```
public class Enquiry extends org.davic.net.ca.Text {
  public String getText();

  public short getAnswerLength();
  public boolean getBlindAnswer();

  public void setAnswer(String answer);

  public void close();
}
```

A `getText()` metódus segítségével kiírathatjuk a kérdéshez szükséges szövegeket. A `getAnswerLength()`beállítja a kérdésre várt válasz hosszát (például PIN kód ellenőrzés esetén). A `setAnswer()`-el az alkalmazás értesíti a CA-rendszert a felhasználó által adott válasz értékéről.

Amikor a felhasználó befejezte a dialógust, az alkalmazásnak meg kell hívnia `close()` metódust. A close() metódus értesíti a CA-rendszert a dialógus befejezéséről. Ezután két dolog következhet, és mindkét lehetséges esetben a CA-rendszernek megvan az összes információja, amire szüksége van. Ekkor vagy a `CloseMMIEvent` elküldésével értesíti az alkalmazást, hogy az interakció véget ért, vagy a `StartMMIEvent` elküldésével elindítja az alkalmazásnak szükséges következő dialógust.

5.4. Feltételes hozzáférés smart card-al

A legtöbb STB manapság már rendelkezik smart card olvasó egységgel is, így ha a smart card-on például digitális aláírás is van letárolva, általa a feltételes hozzáférés egyszerűen megvalósítható. A smart card-ra más elnevezéseket is szoktak használni: chipkártya, mikroprocesszoros kártya, intelligens kártya.

A smart kártyáknak nagyon sok alkalmazási lehetősége van, használják például: elektronikus igazolványként, digitális aláírás tárolására, hozzáférés védelemre, hitelkártyaként, elektronikus pénztárcaként, a SIM-kártya is egyfajta intelligens kártya,

parkoló vagy közlekedési kártyaként, hűségkártyaként és payTV alkalmazások esetén.

Az interaktív alkalmazások szempontjából, a smart card két lényegesebb funkcionalitása, bizonyára a következő:

1. a felhasználók authentikációja és hozzáférési jogosultságainak kezelése, továbbá az alkalmazások biztonságos használata (pl.: T-health alkalmazás esetén betegadatok védelme, T-government alkalmazásoknál az elektronikus igazolvány használata smart card-ként);

2. az egyszerű fizetési lehetőség (pl.: payTV, T-commerce alkalmazás).

Ahhoz, hogy egy MHP alkalmazáshoz használhassuk a smart card-ot, a STB-nak rendelkeznie kell smart card olvasó egységgel, és a middleware-nek implementálnia kell a smart card-ra vonatkozó API-t. Az MHP 1.0.2 [ ] nem követeli meg a smart-card API implementációját, tehát az MHP 1.0.2. middleware-el rendelkező STB-kon nem lehet smart card-ot használni. Viszont, az MHP 1.1-ben [ ] már benne van az OpenCard Framework (OCF) for Embedded Devices 1.1.2. [22], mely szabvány, a smart card-ra épülő alkalmazás fejlesztések egyszerűsítésére lett létrehozva.

Az MHP 1.1.2.-től azonban az OpenCard Framework használata már nem tanácsolt, „halott” specifikációnak tekintik. Az OCF-et a mobiltelefonos fejlesztések Java API-ja váltja fel: a SATSA-APDU. A SATSA-APDU API alacsony szintű protokollt használva biztosítja az alkalmazások kommunikációját a smart card-al.

6. Interaktív szolgáltatás fejlesztése DVB-MHP platformra

Miközben Magyarországon jelenleg még gyerekcipőben jár a földfelszíni digitális televíziózás technológiája, több európai országban már évek óta folynak a fejlesztések. A DVB-T technológiában előrehaladott országok között található Olaszország is, ahol 2008. december 31.-én fognak utoljára analóg adást sugározni, és ezt a dátumot követően teljesen átállnak a DTT-re (Digital Terestrial Televison). A technológia éllovasai közül azért emeltem ki Olaszországot, mert ebben a fejezetben, egy olaszországi cég megbízására készülő, interaktív szolgáltatás bemutatását, a fejlesztés menetének és a fejlesztéshez fűződő, ez idáig felmerült, saját tapasztalataimnak, illetve fejlesztőcsapatunk tapasztalatainak leírását tűztem ki célul. Emellett, jelen fejezetben, az előbb említett szolgáltatáson keresztül szeretnék néhány irányvonalat adni a DVB-MHP platform interaktív szolgáltatásainak fejlesztéséhez.

6.1. Az alapprobléma

A projekt, melynek tervezésében és fejlesztésében részt vettem, egy olyan egészségügyi nyilvántartó rendszer fejlesztését kapta célul, melynek középpontjában egy T-health alkalmazás áll. A rendszer által nyújtott szolgáltatás lényegében, az Olaszországban használatos, úgynevezett „Gyermek-egészségügyi kiskönyv” elektronikus változatából áll.

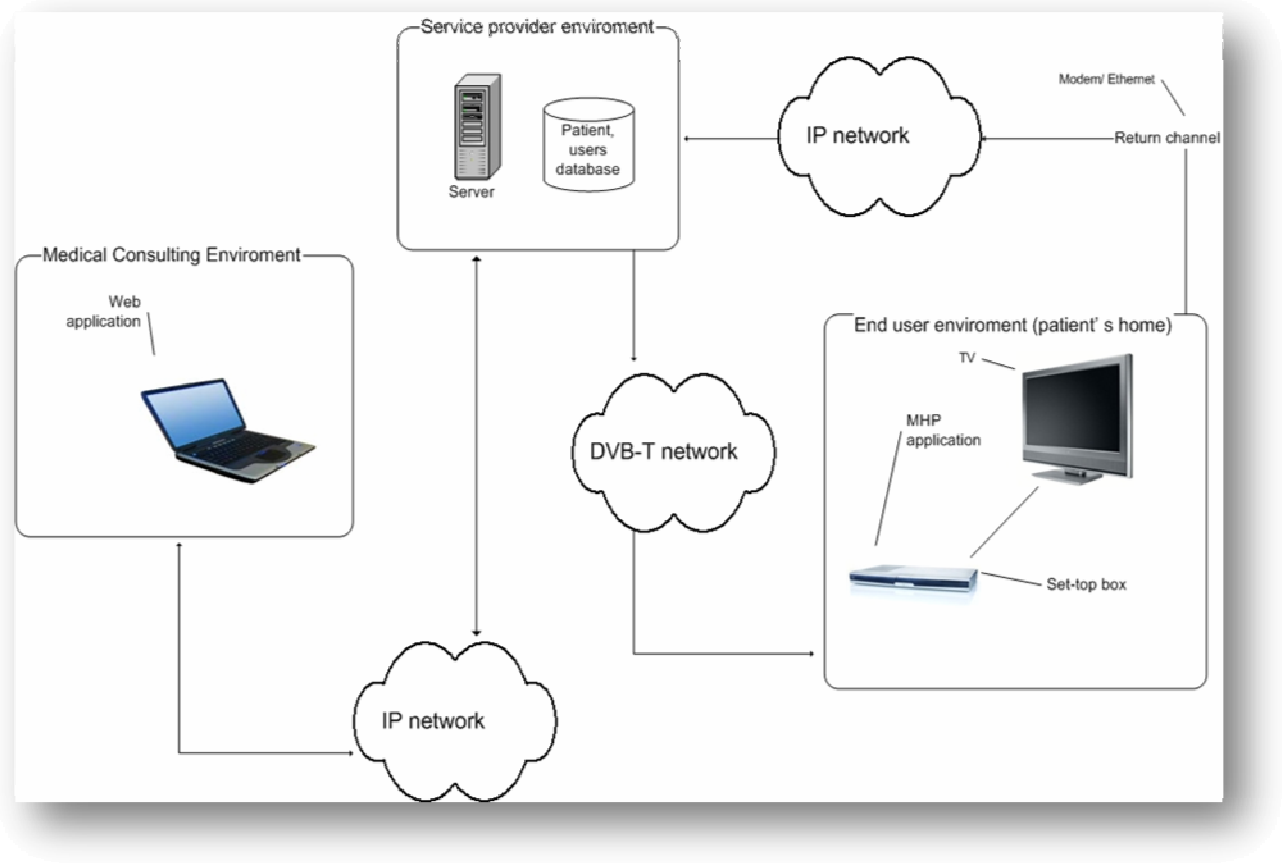
A „Gyermek-egészségügyi kiskönyv”, egy olyan dokumentum, melyben a 0 és 14 év közötti gyermekek egészségére, valamint fejlődésére vonatkozó információkat tárolnak. Olyan adatok nyilvántartására szolgál, mint, a gyermek kórtörténete, anyakönyvi adatok, oltások, betegségek, gyógyszerallergia, beutalások, orvosi vizitek, időszakos egészségügyi mérlegek stb.

A T-health alkalmazásnak a gyermek-egészségügyi könyv megtekintése mellett, olyan interaktivitást is igénylő szolgáltatásokat kell nyújtania, mint: az on-line kérdőívek kitöltése, laboreredmények on-line lekérdezése, smart card-os authentikáció.

6.2. A rendszer architektúrája és funkcionális követelményei

A gyermek-egészségügyi könyv elektronikus változatát az orvosok, a rendelőben egy webalkalmazás formájában érik el, a szülők, pedig otthonról, TV-n keresztül egy MHP alkalmazás formájában. Tehát a rendszer logikailag két modulból áll: egy MHP alkalmazás és egy webalkalmazás.

A két modul jogosultságrendszere, hozzáférési módja és felhasználói felülete nagyban különböznek egymástól. Az orvos karbantarthatja a rendelő és a betegek adatait, feltölthet hasznos dokumentumokat és kérdőíveket, kinyomtathatja az egészségügyi kiskönyvet. Az orvosi rendelőben több jogosultságkör is létezik: orvos, asszisztens és adminisztrátor-karbantartó. A szülő, otthonról, csak megtekintheti az elektronikus könyvet, megjegyzéseket írhat bele (melyeket az orvos nem lát), kérdőíveket tölthet ki, de nem módosíthatja a gyermek egészségére és fejlődésére vonatkozó adatokat, mivel ahhoz csak az orvosnak lehet joga.



9. ábra: A T-health rendszer architektúrája

6.3. Fejlesztői és teszt környezetek

Az MHP-alkalmazás fejlesztői környezetének kiválasztásakor több lehetőségünk is van, azonban ezek között van két egyszerűbb és kényelmesebb lehetőség: az egyik a development set-top box, a másik egy emulátor.

Ha fent említett két egyszerűbb megoldás közül választunk, a fejlesztői környezet a következőkből épül föl:

- egy osztályhalmaz, melynek segítségével lefordíthatjuk az MHP-alkalmazást
- egy szövegszerkesztő és egy fordító (pl. Eclipse) vagy egy MHP authoring eszköz
- egy platform, mellyel futtathatjuk az alkalmazást. Ez lehet egy PC-alapú emulátor, vagy egy development STB.

Egy MHP-alkalmazás fejlesztéséhez és teszteléséhez nem elég megvenni egy egyszerű háztartási használatra szánt STB-ot. Egy developer STB-ra van szükségünk. Egyes gyártók készítenek STB-jaikhoz fejlesztői változatot is, ezek a developer STB-ok. Például az ADB Global-nak [21] is van egy developer STB-ja, egy DTT/IPTV hibrid készülék, mely 1000 dolláros áron van. A másik lehetőség egy emulátor használata, mellyel számítógépen tudjuk futtatni az MHP-alkalmazást. Az emulátorokat elég borsos áron adják (akár 9000 euro), igaz léteznek ingyenes emulátorok is (Xletview [14], OpenMHP [15]).

A hagyományos és a fejlesztői STB-ok közötti két legfőbb különbség közül az egyik a debug-konzol létezése, ahová az alkalmazás üzeneteket ír ki, arról, hogy éppen mi történik. Ez történhet egy soros- vagy egy Ethernet kapcsolaton keresztül. A másik különbség az alkalmazások letöltése a set-top box-ra, transport stream használata nélkül. Egyes STB-ok ezt egy soros kapcsolaton keresztül valósítják meg, más készülékek TFTP-t használnak Ethernet kapcsolaton keresztül, de léteznek olyan készülékek is, melyek saját merevlemezére lehet letölteni az alkalmazásokat.

A fizetős emulátorok sok olyan funkcióval rendelkeznek, melyekkel szinte teljesen kiküszöbölhetők a hagyományos PC-s alkalmazások és az MHP-alkalmazások közötti

különbségek. Ha ilyen emulátort használunk, főleg olyat, amelyben designer eszköz is van, kevés MHP ismeret is elég ahhoz, hogy fejlesszünk.

Az ingyenes emulátorok, például az XleTView vagy az OpenMHP, jól használhatók arra, hogy megtanuljunk egyszerű MHP-alkalmazásokat fejleszteni. Egy projekt kezdeti fázisában még megteszi, de később semmiképp sem javasolt a használatuk, mert nagyon sok fontos funkcionalitás hiányzik belőlük, és sokszor hibásan is működnek.

Fontos megjegyezni, hogy, noha az egyes fizetős emulátorok nagyon jól szimulálják az alkalmazások tényleges működését egy valódi STB-on, a hardverkülönbözőségek miatt mégsem lehet teljesen rájuk hagyatkozni. Emulátort használva, amellett, hogy nagyon nehéz megmondani, milyen gyors lesz az alkalmazás és milyen gyorsan fog letöltődni a STB-ra, az alkalmazást mégis csak a számítógép képernyőjén látjuk és nem egy TV készülék képernyőjén, melyen a grafikus interfész, ahogy azt a 4. fejezetben a grafikus modellnél már tárgyaltuk, valószínű máshogyan fog kinézni.

6.3.1. Emulátorok és authoring eszközök

Egy MHP-alkalmazás szerkesztésére és fordítására használhatunk egy hagyományos szövegszerkesztőt és egy Java-fordítót, vagy egy authoring eszközt. Az authoring eszköz használata nagyon megkönnyíti azok dolgát, akik egyáltalán nem, vagy csak kevésbé jártasak az MHP-s fejlesztésekben. Egyesek viszont, főleg a generált kód minősége miatt, nem szeretik ezt a megoldást és inkább a hagyományos Java fejlesztői környezeteket választják, mint az Eclipse vagy a JBuilder.

Az authoring eszközök, általában tartalmazzák azokat az osztályokat, melyek szükségesek egy MHP-alkalmazás fordításához. Az AltiComposer (Alticast), a Cardinal Studio (Cardinal InformationSystems), a JAME (Fraunhofer IMK), az iTV Suite(Icareus), a Pontegra (Nionex), a Set Media (Interattiva) stb. mind olyan authoring eszközök (egyesek közülük egyben emulátort is), melyeket MHP-alkalmazások szerkesztésére és fordítására alkottak. Segítségükkel, nagyon egyszerűen készíthető el az alkalmazás felhasználói interfésze és a navigáció is. Egyes authoring eszközök, olyan szolgáltatásokat is kínálnak, mint az alkalmazás elemzése, tesztelése és aláírása.

Miért és milyen esetekben érdemes authoring eszközt használni?

Az authoring eszközök előnyeiként sorolhatjuk fel a következőket: használatuk jelentősen csökkenti a fejlesztési időt, a fejlesztéshez kevesebb Java/MHP tudás is elég, könnyen és hatékonyan tesztelhetjük az alkalmazásokat, és egyszerűen implementálhatjuk a változásokat. Az authoring eszközök használata a következő típusú alkalmazások fejlesztéséhez javasolt: hírportálok, news ticker alkalmazások, időjárás-előrejelző, interaktív műsorok (interaktív játékok, szavazások), egyszerű játékok (Pl. Tetris).

Mikor nem javasolt az authoring eszközök használata?

Akkor, ha nagyon kis méretű az alkalmazás (kisebb mint 100 kByte), ha bonyolult folyamatokat kell implementálnunk, melyek ráadásul gyors futást is igényelnek és legfőbbképpen akkor, ha tapasztalt Java/MHP fejlesztők vagyunk, mert akkor valószínű, hogy rövidebb, hatékonyabb és gyorsabb kódot tudunk írni. Például böngésző, feliratozó alkalmazás vagy számlázó alkalmazás fejlesztése esetén javasolt authoring eszköz használata helyett, a hagyományos szerkesztés és fordítás.

6.3.2. Távoli tesztkörnyezetek

A fejlesztés során azzal is számolnunk kell, hogy a különböző STB-kon, különböző módon futhat az alkalmazás, mivel az MHP-middleware implementációja gyártónként változhat. Ezért jó minél több STB-n kipróbálni, legyen az sima háztartási vagy fejlesztői verzió.

Ha nem tudunk beruházni egy saját STB-farm kialakítására, használhatunk távoli online tesztkörnyezeteket is, ahol ki lehet próbálni az alkalmazást több típusú STB-on, az MHP szabvány különböző verzióival (MHP 1.0.2, MHP 1.0.3., MHP, 1.1.1). Ilyen például az **IRT-Testcentre** is.

Valamelyik online tesztkörnyezetben való teszteléshez, regisztrálnunk kell és aztán előzetes foglalást kell tennünk az MHP-KDB portálon [23]. Jelen szabályok szerint, a foglalás a kívánt időpont előtt két hónappal történhet, és minimum 15 perces, maximum 180 perces időszeletet tudunk magunknak lefoglalni. A lefoglalt időtartamban kizárólagos hozzáférésünk lesz az online tesztkörnyezethez.

A következő kép az IRT-Testcentre-ből származik. A képen látszik, hogy ki lehet választani a STB típusát. Ha az ki lett választva, legalul megjelenik a kiválasztott STB típusa, hardver és szoftver információk és megjelennek a STB-hoz tartozó távirányító gombjai is.

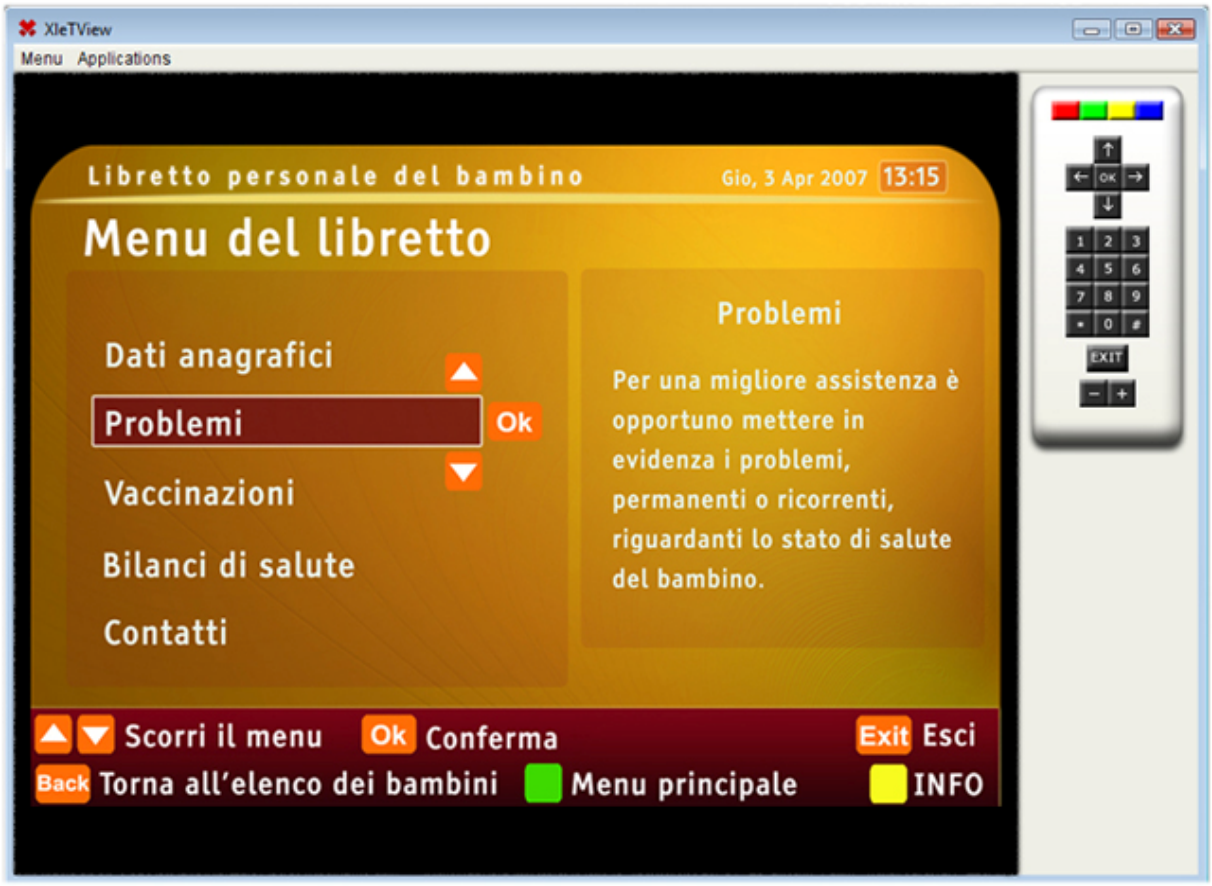
10. ábra: Távoli tesztkörnyezet. Forrás: IRT-Testcentre

6.3.3. A T-health alkalmazás fejlesztői környezete

Mivel fejlesztőcsatunkból ez idáig még senki sem dolgozott MHP-projekten, úgy döntöttünk, hogy a fejlesztés korai fázisában egy ingyenes emulátort fogunk használni. Kipróbáltuk az OpenMHP-t is, de végül az XletTView mellett döntöttünk, mert az jobbnak tűnt, a dokumentáció több volt hozzá és az MHP-fórumokon is többet foglalkoztak vele.

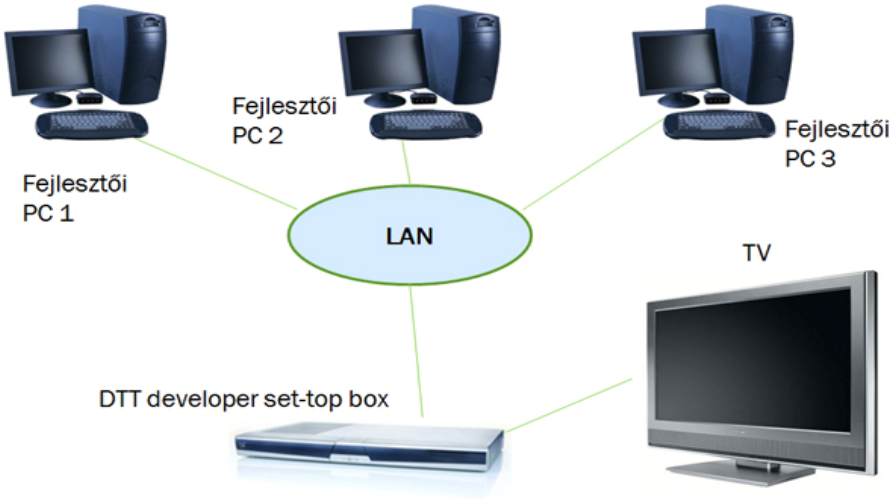
Az XleTView egy ingyen letölthető emulátor, mely számítógépen emulálja az MHP-alkalmazások futtatásához szükséges környezetet. Az XleTView nem teljes MHP implementáció, ezért sok funkcionalitás hiányzik belőle. Szintén az implementáció hiányosságai miatt, bizonyos esetekben az alkalmazás máshogyan viselkedhet, mint ha egy valódi MHP-implementációt használnánk. Ezek miatt, komoly projektek esetében, az

XleTView-ra tényleg csak átmeneti megoldásként tekinthetünk.



11. ábra: Az alkalmazás egyik menüje XleTView-ban

Az XleTView-t használtuk arra, hogy összeállítsuk az alkalmazás vázát és arra, hogy az MHP technológiából adódó akadályokat megismerjük és kiküszöböljük (például a válasz csatorna vagy a megjelenítés problémáját). Miután elkészítettünk pár oldalt és megoldottuk a kommunikációt az adatbázissal és a szerverrel, áttértünk a developer set-top box használatára. Ugyanis egy authoring eszköz és egy emulátor megvásárlása helyett, egy ADB 3800TW típusú fejlesztői set-top box-ot rendeltünk. Ez egy hibrid STB, mely képes egyaránt DTT és IPTV alkalmazások futtatására. A dobozon Linux operációs rendszer fut, MHP 1.1.2. middleware van rajta, képes a JavaScript-ek kezelésére és van rajta Mozilla böngésző.



12. ábra: A T-health alkalmazás fejlesztői környezete

A Java kód szerkesztésére és fordítására az Eclipse fejlesztőkörnyezetet használtuk, adatbázisként MySQL-t, adatbázis-lekérdezőként, pedig a Hibernate-t.

6.4. Grafikus felhasználói felület és navigáció

Egy interaktív digitális TV-s (iTV) platformon futó alkalmazás fejlesztésének első lépése egy könnyen használható felhasználói interfész elkészítése. Erre a problémára nem készült szabvány, csak ajánlások léteznek. A T-health alkalmazásunk grafikus interfészének megtervezésekor főleg az Ugo Bordoni Alapítvány [9] ajánlásaira támaszkodtunk.

6.4.1. Az alkalmazás elérése és kontextusa

Egy interaktív alkalmazás jelenlétét, a felhasználó számára világosan jelezni kell, a hozzáférésnek egyértelműnek és egyszerűnek kell lennie. Például ha hozzáférhető egy olyan alkalmazás, mely sporteseményeket érintő információkat szolgáltat, miközben a felhasználó passzívan tévézik a következő üzenettel, értesíthetjük az alkalmazás elérhetőségéről: „A sportesemények megjelenítéséhez nyomja meg a  gombot!”

Az interaktív alkalmazás eléréséhez és a kilépéshez használandó gombok

Ahogyan már említettem, nem létezik konkrét szabvány a felhasználói interfész kialakítására

vonatkozóan, ám sok esetben a sugárzók ugyanazokat a megoldásokat használják. A műsorterjesztők többségénél, az interaktív tartalom eléréséhez, általában a piros gombot kell megnyomni. Kötelező feltüntetni egy gombot (legtöbbször ez az EXIT gomb), mellyel kiléphetünk az alkalmazásból és visszatérhetünk a TV csatornához.

Minimize/maximize gomb

Tanácsolt egy gomb kijelölése az alkalmazások ikonná minimize-olására, így anélkül hogy kilépnénk az alkalmazásból, nézhetjük a TV-műsorokat is.

Üzenetek

Ugyanúgy ahogyan a PC-s alkalmazások esetén, itt is javasolt tájékoztatni a felhasználót arról, hogy éppen mi történik a képernyőn megjelenő üzenetek formájában (például tájékoztató üzenet egy alkalmazás betöltésének állásáról).

6.4.2. Navigáció

Az interaktív alkalmazások navigációját jól át kell gondolni, mivel billentyűzet helyett, itt csak egy távirányító áll rendelkezésünkre.

A tartalmak hierarchiája - menük

A menü elkészítésekor érdemes odafigyelni a tartalmak csoportosítására: legyen logikus és egyszerűen megjegyezhető az elérés. Fontos, hogy a tartalom hierarchiájának ne legyen háromnál több szintje, egyébként könnyen eltévedhetünk a menüben.

Navigáció a menükben

A menükben való navigációt a következő módszerek valamelyikével érdemes megoldani:

1. A le, fel, jobbra, balra nyilakat használva pozícionálunk a menüpontokra, kiválasztani, pedig az OK/ENTER/SELECT gombbal lehet (a távirányító gombkészlete STB-onként változhat).

2. A numerikus gombot hozzárendelése az egyes menüpontokhoz. Ez a megoldás nyilván akkor ésszerű, ha a menüpontok száma kevesebb tíznél.

3. A színes gombok hozzárendelése a fő funkciókhoz.

A T-health alkalmazásunk navigációját úgy alakítottam ki, hogy a képernyőn egyszerre csak egy szint szerepeljen a menürendszer hierarchiájából és az előző szinthez, ha van olyan, pedig a piros gomb megnyomásával lehessen visszatérni (lásd a **11. ábrát**). Ha a navigációs hierarchia több szintjét is megjelenítjük a képernyőn, főleg az egér hiánya miatt, nehézkessé válik a menürendszer bejárása.

Vezérlő gombok

Ami még a navigációt illeti, az Ugo Bordoni Alapítvány [9] azt javasolja, hogy vezérlő gombokat (↓,↑,←,→, OK, BACK, EXIT stb.) vízszintesen helyezzük el, és a színes gombokat lehetőleg mindig ugyanabban a sorrendben tüntessük fel.

6.4.3. Look & Feel

Az interaktív MHP-alkalmazások grafikus interfészének tervezésekor, figyelembe kell venni a technológia sajátosságait a képek és a szövegek megjelenítését illetően is.

Font típusa

Nyilvánvaló, hogy a felhasználók a televíziót nagyobb távolságról fogják nézni, mint egy számítógép képernyőjét. Tehát ahhoz, hogy garantálni tudjuk az olvashatóságot minimum 20-as maximum 36-os (címek esetében) betűmérettel kell kiírni a szövegeket. A 24/26-os mérettel írott Tiresias típusú font jól használható és körülbelül ugyanúgy mutat a monitoron, mint a tévéképernyőn.

Villogás (flickering)

A villogó effektus valójában az, amikor szabad szemmel látható, ahogy egy pixel megjelenik és eltűnik a képernyőről. Ez a váltósoros (interlace) képernyő miatt fordul elő, akkor, ha 1 pixelnél vékonyabb vonalat húzunk a képernyőn. Fontos tehát, például keretek kirajzolásánál, minimum 3 pixel vastagságú vonalakat húzni, mert így nehezebben észlelhető a villogás. A másik ok, amiért javasolt vastagabb vonalakat húzni, a pixelek szögletes, és nem négyzet alakja miatt, ami enyhe vízszintes torzuláshoz vezet.

Színek használata

A színek kiválasztásánál, oda kell figyelnünk arra, hogy a televízió készülék által használt színskála szűkebb, mint a számítógépek esetében. A DVB-MHP szabvány 188 színre

korlátozta a színpalettát, és nekünk ehhez kell igazodnunk. Szintén a villogás elkerülése miatt, nem javasolt a fehér és a piros szín használata (természetesen ez a szín dominanciára értendő).

6.4.4. Szövegek megjelenítése

Ami a szövegek megjelenítését illeti, a olvasás megkönnyítése érdekében jó néhány dologra oda kell figyelnünk. A felhasználó, aki megszokta, hogy tévénézés közben az interaktivitás a gombok nyomkodásában merül ki, nem fog túlzottan koncentrálni a képernyőn megjelenő szövegre. Ezért a tartalmat úgy kell szervezni, hogy az ne terhelje le túlzottan a felhasználót.

Szövegmegjelenítés

Javasolt a fontos információkat a szöveg elejére tenni. Nem javasolt egyszerre 5 sornál hosszabb szöveget kiírni, továbbá a hosszú szövegeket üres sorokkal kell tagolni. Érdemes vázlatszerűen, pontozott listában megjeleníteni a szöveget.

Lapozás

Ha a szöveg meghaladja a képernyő méretét, érdemes *scroll bar*-t használni, melyet a fel és le nyilakkal lehet vezérelni. A *page up*, *page down*-t a jobbra, balra nyilakkal javasolt megoldani.

Színek szemantikus használata

Hasznos figyelmet szentelni a színek jelentésére is, ugyanis a színek általában érzéseket váltanak ki. Például a piros veszély jelent, a sárga figyelmet, a fehér és a szürke semlegességet. Azzal is számolnunk kell, hogy a színek jelentése az egyes kultúrákban mást és mást jelenthet. A színek szemantikus használata, főleg a távirányító színes gombjainak használatára vonatkozóan lehet fontos.

7. Összefoglalás

A diplomamunka, amellett hogy kitekintést nyújt a manapság oly divatos és feltörekvő, digitális televíziózás interaktív világába, az MHP alkalmazások fejlesztésének elkezdéséhez szükséges tudnivalókat ismerteti. A dolgozatnak nem volt célja a digitális tévéhez kapcsolódó technológiák és szabványok részletes áttekintése, pusztán a gyakorlat kívánta mélységben mutattam be őket.

A dolgozat írása közben, törekedtem minél szélesebb látókört adni a témára vonatkozóan, ám terjedelmi okokból, csak a legfontosabb tudnivalókat sikerült részleteznem. A digitális televíziózás világát informatikai, de leginkább alkalmazásfejlesztői szempontból közelítettem meg. Ezért a dolgozat csak áttekintést nyújt a legfontosabb kapcsolódó szabványokról, a DVB és az MHP-ről, és középpontjában a Java alapú, interaktív MHP szolgáltatások fejlesztése áll. Ismertettem az MHP alkalmazás felépítését és működését, illetve a fejlesztéshez használható Java eszközrendszereket, API-kat, miközben kitértem a digitális televíziós platform és technológia adta, olyan különleges kérdésekre is, mint az alkalmazás grafikus és biztonsági modellje.

A T-health alkalmazás bemutatásával, megpróbáltam néhány irányvonalat adni az MHP-s fejlesztések elkezdéséhez. A T-health alkalmazás ismertetése során, a célom az volt, hogy egy konkrét példán keresztül szemléltessem az MHP alkalmazások fejlesztését, ám a probléma tárgyalását, csak olyan mélységig szándékoztam elvinni, ameddig a technológia adta akadályok el nem hárulnak. Emiatt tartottam fontosnak kitérni a konkrét T-health rendszer architektúrájára, a fejlesztői és a teszt környezetre, a hardver és szoftver követelményekre és a felhasználói interfész elkészítésére.

A diplomamunkám témája, jelenleg szinte érintetlen területnek számít Magyarországon, elkészítésében szinte kizárólag külföldi irodalomra tudtam támaszkodni. Az egy főre jutó napi átlagos 4-4,5 órás tévézéssel, a világ élmezőnyében feszítő Magyarországon, úgy gondolom, hamarosan nagyon meg fog nőni a téma iránti érdeklődés.

Ilyen statisztikák mellett, nagy horderejű kérdés, hogy a jövőben miként alakul át a televíziós szolgáltatók piaca. Annyi azonban biztos, hogy hatalmas üzleti lehetőség rejlik az

analógról a digitális televíziózásra történő átállásban. Az hogy a földfelszíni, az internetes (IPTV), a kábeles, a műholdas és a mobil tévé közül melyik digitális technológia kerül ki győztesnek a tévépiac átrendeződése során, az egyelőre még kérdéses, de a jövő majdnem biztos, hogy sok platform egymás melletti párhuzamos működését fogja hozni. Emiatt nagyon fontos, hogy többplatformos, vagy platformfüggetlen interaktív alkalmazásokat lehessen fejleszteni. Ha az ETSI jóváhagyja az IPTV-re vonatkozó DVB-IP szabványt, olyan interaktív alkalmazások fejlesztésére nyílik lehetőség, melyek négy digitális technológiától függetlenek: a DVB-T, a DVB-C, a DVB-S és végül az IPTV-től.

Irodalomjegyzék

1] Steve Morris and Anthony Smith-Chaignea: *Interactive TV standards: A Guide to MHP, OCAP, and JavaTV*, Focal Press, April 2005

[2] ETSI: TS 201 812 V1.1.1: *Digital Video Broadcasting, Multimedia Home Platform Specification 1.0.3*, http://www.etsi.org

[3] Gian Paolo Balboni, Giovani Venuti, DTT e servizi interattivi: *Come e perche della nuova televisione*, Telecom Italia LAB

[4] DVB: *Digital Video Broadcasting*, www.dvb.org

[5] MHP: *Multimedia Home Platform*, www.mhp.org

[6] HAVi: *Home Audio/Video Interoperatibility*, http://havi.org

[7] http://www.tvwithoutborders.com/tutorials/getting_started

[8] http://www.tvwithoutborders.com/tutorials/mhp

[9] Fondazione Ugo Bordoni: *Raccomandazioni per le interfacce dei servizi interattivi della televisione digitale*

[10] Steve Morris: *Mhp vs. Java, The Differences*, http://www.interactivetvweb.org/resources/presentations/mhpjavadifferences.ppt

[11] ETSI: *European Telecommunication Standards Institute*, http://www.etsi.org

[12] Java TV: http://java.sun.com/products/javatv/

[13] Personal Basis Profile: http://java.sun.com/products/personalbasis/

[14] XletView: http://xletview.sourceforge.net/

[15] OpenMHP: http://www.openmhp.org

[16] *MHP guide*: http://www.mhp-knowledgebase.org/publ/mhp-guide.pdf

[17] Tim Koch, Usability_and_Style_Guide.pdf

[18] *The OEM PersonalJava Application Environment Version 1.2a specification* - http://java.sun.com/products/specformhp

[19] G.P.Balboni, G.Venuti, *Digital Teresstrial Television e servizi interattivi*, Edizioni Tilab 2005

[20] *Il progetto Digital Video Broadcasting* www.telecomitalialab.com

[21] ADB, Advanced Digital Broadcaster, www.adbglobal.com

[22] *OpenCard Framework for Embedded Devices Specification*, IBM

[23] *MHP knowledgebase*: http://www.mhp-knowledgebase.org/

Függelék

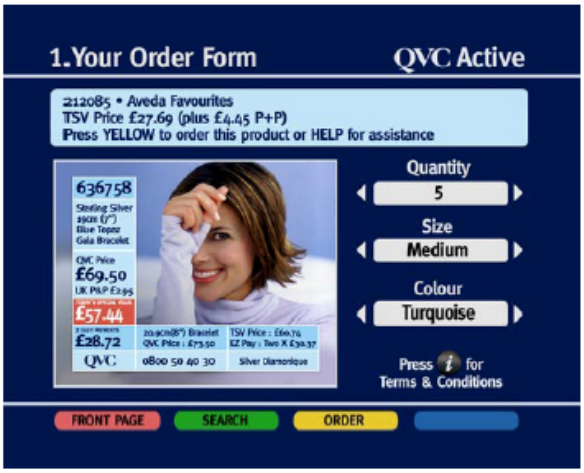
1. Függelék: néhány példa interaktív MHP alkalmazásokra

Hírek, időjárás

Szórakozás, játékok

Sport és fogadások

T-Commerce és iADV (i-advertisment)



EPG és T-government

2. Függelék : MHP-alkalmazás jogosultság kéréséit tartalmazó XML állomány példa

```
<?xml version="1.0"?>
<!DOCTYPE permissionrequestfile
  PUBLIC "-//DVB//DTD Permission Request File 1.0//EN"
  "http://www.dvb.org/mhp/dtd/permissionrequestfile-1-0.dtd">

<permissionrequestfile
  orgid="0x000023d2"
  appid="0x0020">

  <file value="true"></file>

  <capermission>
    <casystemid
      id="0x1111" messagepassing="true"
      entitlementquery="true" mmi="false">
    </casystemid>
  </capermission>

  <applifecyclecontrol
    value="true">
  </applifecyclecontrol>

  <returnchannel>
    <defaultisp></defaultisp>
    <phonenumber>+3583111111</phonenumber>
    <phonenumber>+3583111112</phonenumber>
    <phonenumber></phonenumber>
  </returnchannel>

  <tuning value="false"></tuning>
  <servicesel value="true"></servicesel>
  <userpreferences
    read="true"
    write="false">
  </userpreferences>

  <network>
    <host action="connect">hostname</host>
  </network>
```

```
  <persistentfilecredential>
    <grantoridentifier id="0x0202030">
    </grantoridentifier>
    <expirationdate date="24/12/2032">
    </expirationdate>
    <filename read="true" write="false">
      5/15/dir1/scores
       </filename>
    <filename read="true" write="false">
      5/15/dir1/names
    </filename>
    <signature>
      02320329329293293292149314392942394329423943 2
    </signature>
    <certchainfileid>3</certchainfileid>
  </persistentfilecredential>

</permissionrequestfile>
```