UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA U NOVOM SADU

Nemanja Lukić

Predlog proširenja Android operativnog sistema servisima digitalne televizije

DOKTORSKA DISERTACIJA

Mentor:
prof. dr Miodrag Temerinac

Novi Sad, 2014.

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		Монографска документација
Тип записа, **ТЗ**:		Текстуални штампани материјал
Врста рада, **ВР**:		Докторски рад
Аутор, **АУ**:		Немања Лукић, дипл. инж.
Ментор, **МН**:		проф. др Миодраг Темеринац
Наслов рада, **НР**:		Предлог проширења Андроид оперативног система сервисима дигиталне телевизије
Језик публикације, **ЈП**:		Српски / латиница
Језик извода, **ЈИ**:		Српски
Земља публиковања, **ЗП**:		Република Србија
Уже географско подручје, **УГП**:		Војводина
Година, **ГО**:		2014.
Издавач, **ИЗ**:		Ауторски репринт
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		8 поглавља / 181 страна / 104 цитата / 29 табела / 73 слике / 2 прилога
Научна област, **НО**:		Електротехника и рачунарство
Научна дисциплина, **НД**:		Рачунарска техника
Предметна одредница/Кључне речи, **ПО**:		Андроид оперативни систем, дигитална телевизија, Јава, програмска спрега, сервиси дигиталне телевизије, уграђене платформе
УДК		
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		Ова дисертација се бави истраживањем у области интеграције сервиса дигиталне телевизије у модерне уређаје потрошачке електронике. Циљ тезе је да развије приступ за системско проширење Андроид оперативног система сервисима дигиталне телевизије, и да предложи решење које омогућује рад у реалном времену. Квалитет решења се оцењује одговарајућим метрикама преко оцене квалитета имплементиране Јава објектно оријентисане спреге за ТВ сервисе. Основни допринос тезе се огледа у дефинисању јединствене програмске спреге сервиса дигиталне телевизије на платформама које прате парадигму виртуелне машине. Предложено решење омогућује развој апликација оптимизованих за извршавање на ТВ уређајима и даље спрезање података ТВ сервиса са остатком Андроид екосистема.
Датум прихватања теме, **ДП**:		29.05.2014.
Датум одбране, **ДО**:		
Чланови комисије, **КО**:	Председник:	др Никола Теслић, ред. проф.
	Члан:	др Јован Ђорђевић, ред. проф.
	Члан:	др Иштван Пап, доцент
	Члан:	др Милан Видаковић, ванред. проф.
	Члан, ментор:	др Миодраг Темеринац, ред. проф.

Потпис ментора

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Doctoral dissertation	
Author, **AU**:		Nemanja Lukić	
Mentor, **MN**:		Miodrag Temerinac, PhD	
Title, **TI**:		One approach to the extension of Android operating system with digital TV services	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Republic of Serbia	
Locality of publication, **LP**:		Vojvodina	
Publication year, **PY**:		2014.	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		8 chapters / 181 pages/ 104 references / 29 tables / 73 pictures / 2 appendixes	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		Android operating system, digital television, Java, digital television services, real-time embedded platforms	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		This PhD dissertation addresses the problem of integration of the digital TV services inside modern consumer electronic devices. The main focus of the dissertation is a development of systematic approach for extension of Android operating system with support for digital television. Combined with this, the dissertation describes solution in form of hardware platform with accompanying software that closely follows this approach and achieves real-time performance. Quality of proposed solution is benchmarked using metrics for measuring quality of object-oriented program code. The main contribution of the dissertation is unification of system software API for digital television on Android-based platforms. Proposed solution allows development of TV-centric software capable of real-time performance, and further native integration of data coming from DVB broadcast into Android ecosystem.	
Accepted by the Scientific Board on, **ASB**:		29.05.2014.	
Defended on, **DE**:			
Defended Board, **DB**:	President:	dr Nikola Teslić, Professor	
	Member:	dr Jovan Đorđević, Professor	
	Member:	dr Ištvan Papp, Assistant Professor	Menthor's sign
	Member:	dr Milan Vidaković, Associate Professor	
	Member, Mentor:	dr Miodrag Temerinac, Professor	

Sažetak

Trenutno stanje potrošačke elektronike beleži pojavu jedinstvenih digitalnih servisa koji su korisniku raspoloživi kroz različite uređaje, od televizijskih prijemnika do mobilnih uređaja kao što su pametni telefoni ili tableti. Ta pojava je omogućena kroz konvergenciju između računarskih mreža, mobilnih uređaja, ali i same digitalne televizije. Ovu pojavu prati povećanje procesne moći ugrađenih arhitektura koje danas omogućuju razvoj modernih platformi sposobnih da izvrše kompleksne aplikacije pisane u programskim jezicima koji koriste paradigmu virtuelne mašine. Osim ovog trenda, primetno je da proizvođači TV uređaja sve više koriste Android operativni sistem, kao bazu za svoja rešenja. Odabir ovog operativnog sistema proizvođačima omogućuje da njihova rešenja, osim standardnih TV servisa, pruže korisniku i napredne funkcije, kroz aplikacije koje se mogu preuzeti sa raspoloživog otvorenog marketa i instalirati na krajnji TV uređaj. Mana upotrebe Android operativnog sistema kao programske osnove za TV uređaje, se ogleda u nepostojanju sistemske programske sprege za upravljanje podacima iz digitalnog televizijskog transportnog toka.

Cilj ove disertacije je da definiše predlog za sistemsko proširenje Android operativnog sistema servisima digitalne televizije i ponudi rešenje koje omogućuje rad u realnom vremenu. Kvalitet rešenja je ocenjen odgovarajućim metrikama, preko ocene kvaliteta implementirane Java objektno orijentisane sprege za TV servise. Predloženo rešenje omogućuje razvoj aplikacija optimizovanih za izvršavanje na TV uređajima i dalje sprezanje podataka TV servisa sa ostatkom Android ekosistema.

Abstract

Current trend in the consumer electronics market is the appearance of unique digital services, which users can access through different kinds of available devices, including TV sets, and mobile devices such as smart phones or tablets. This trend is available due to convergence between computer networks, mobile devices and digital television. Together with this, modern embedded architectures provide enough processing power to execute complex applications written in programming languages that use virtual machine paradigm. There is also noticeable increase in usage of Android operating system amongst TV manufacturers. Utilization of this operating system allows manufacturers to provide solutions, which beside standard TV services, provide more advanced functions to users through downloadable applications from available open market. The main flaw of Android operating system, as the base for modern TV platforms, is that it doesn't contain system-level support for digital television services.

The main goal of this dissertation is to develop a systematic approach for the extension of Android operating system with support for digital television, and propose real-time solution on existing hardware platform. Quality of proposed solution is benchmarked using metrics for measuring quality of object-oriented program code and compared with existing solutions on the market. Proposed solution allows development of TV-centric software capable of real-time performance, and further native integration of data coming from DVB broadcast into Android ecosystem.

ZAHVALNOST

Želeo bih da se najiskrenije zahvalim mentoru prof. dr Miodragu Temerincu na strpljenju i motivaciji tokom doktorskih studija, kao i iskrenoj podršci u naučno-istraživačkom radu. Zahvalio bih se i svim članovima komisije na ukazanoj pažnji, konstruktivnim komentarima i suvislim sugestijama.

Posebnu zahvalnost dugujem prof. dr Nikoli Tesliću za iskrenu podršku i inspiraciju prilikom istraživanja koje je rezultovalo ovom disertacijom.

SADRŽAJ

SPISAK SLIKA

SPISAK TABELA

SKRAĆENICE

A/V	*Audio/Video*
AI	*Artificial Intelligence*
ASHMEM	*Anonymous Shared Memory*
ATSC	*Advanced Television Systems Committee*
BCD	*Binary-Coded Decimal*
CAM	*Conditional Access Module*
CES	*Consumer Electronics Show*
CI/CI+	*Common Interface*
CPU	*Central Processor Unit*
CVBS	*Composite Video (Color, Video, Blanking, and Sync)*
DLNA	*Digital Living Network Alliance*
DVB	*Digital Video Broadcast*
DVD	*Digital Video Disc*
DP	*Dynamic Programming*
EIT	*Event Information Tables*
ES	*Elementary Stream*
ETSI	*European Telecommunications Standard Institute*
GPU	*Graphics Processing Unit*
GUI	*Graphical User Interface*
HAL	*Hardware Abstraction Layer*

HbbTV	*Hybrid Broadcast Broadband TV*
HDMI	*High-Definition Multimedia Interface*
HW	*Hardware*
IBC	*International Broadcasting Convention*
IC	*Integrated Circuit*
IP	*Internet Protocol*
IPC	*Inter-Process Communication*
IPTV	*Internet Protocol TeleVision*
JIT	*Just-In-Time*
JNI	*Java Native Interface*
MOOD	*Metrics for Object Oriented Design*
MPEG	*Moving Picture Experts Group*
TS	*Transpor Stream*
TV	*Television*
TVC	TV-centric applications
TV MW	*TV Middleware*
OO	*Object-Oriented*
OS	*Operating System*
PES	*Packetized Elementary Stream*
PID	*Process Identifier*
PiP	*Picture in Picture*
PSI	*Program-Specific Information*
PVR	*Personal Video Recorder*
RGB	*Red-Green-Blue color space (synonym for VGA video interface)*
ROM	*Read-Only Memory*
SDK	*Software Development Kit*
SoC	*System on Chip*
STB	*Set-Top Box*
S-Video	*Separate Video*
TS	*Transport Stream*
URI	*Unified Resource Identifier*
UTC	*Coordinated Universal Time/Universel Coordonné*

USB	*Universal Serial Bus*
VM	*Virtual Machine*
VoIP	*Voice over IP*
ZIP	*Zone Improvement Plan*

POGLAVLJE 1.

UVOD

Polje primenjene elektronike je danas obeleženo ponudom jedinstvenih digitalnih servisa, koji su krajnjem korisniku raspoloživi na različitim uređajima potrošačke elektronike, od televizijskih prijemnika do mobilnih telefona. To je omogućeno kroz konvergenciju između računarskih mreža (Internet), mobilnih uređaja, kao i digitalne televizije.

U isto vreme, sa tehnološkog aspekta, potrošačka elektronika nastavlja da dobija momentum razvojem i upotrebom ugrađenih uređaja sa sve većom procesnom moći. Ovo povećanje se ogleda kako u povećanju radnog takta, tako i u povećanju broja procesorskih jezgara na jednom integrisanom kolu. Ovo za posledicu ima da su danas uređaji potrošačke elektronike u mogućnosti da izvrše kompleksne aplikacije pisane u programskim jezicima koji koriste paradigmu virtuelne mašine.

Većina modernih TV uređaja prati ovaj trend i, osim što korisniku služe za prikaz osnovnih digitalnih TV servisa, nude dovoljnu procesnu moć za izvršavanje naprednih aplikacija pisanih u Java programskom jeziku. Ove aplikacije obezbeđuju korisniku da na TV uređaju pristupa naprednim digitalnim servisima, kao što su pregled socijalnih mreža, čitanje vesti, pristup Internet servisima ili gledanje usluge video na zahtev.

Trenutni trend na polju proizvođača TV uređaja je korišćenje Android operativnog sistema kao baze za svoja rešenja. Android operativni sistem koristi Java

programski jezik kao aplikativno okruženje za razvoj aplikacija. Odabir ovog operativnog sistema proizvođačima omogućava da njihova rešenja, osim standardnih TV servisa, pruže korisniku i napredne funkcije kroz aplikacije koje se mogu preuzeti sa raspoloživog otvorenog marketa aplikacija i instalirati na krajnjem uređaju.

Mana Android operativnog sistema, sa stanovišta razvoja aplikacija za TV uređaje, jeste nepostojanje sistemske programske sprege za upravljanje podacima koji dolaze iz TV prenosa. Ovo sprečava razvoj Android aplikacija koje će implementirati funkcionalnost standardne TV aplikacije. Ova funkcionalnost omogućuje prikaz TV signala korišćenjem standardnih Android video komponenti, dobavljanje i prikaz EPG podataka, teleteksta, prevoda, i sl. Takođe, nepostojanje sistemske podrške za digitalne TV servise sprečava sprezanje naprednih digitalnih servisa prisutnih u Android operativnom sistemu sa informacijama dobijenim iz digitalnog transportnog toka.

Savremena tehnička rešenja, koja integrišu TV servise u Android platforme (ali i druge operativne sisteme bazirane na Java programskom jeziku), zasnivaju se na proširenju Java sistemskog nivoa, programskom spregom specifičnom za TV funkcionalnost.

Cilj disertacije je da istraži moguće pristupe za proširenje Android operativnog sistema servisima digitalne televizije, i da predloži rešenje koje postiže rad u realnom vremenu, zadovoljava zadate zahteve u pogledu složenosti i kompletnosti proširenja, ali je ujedno i nezavisno od platforme na kojoj se izvršava. Ovo rešenje će predstavljati operativni sistem za digitalne TV uređaje koji omogućuje implementaciju različitih digitalnih servisa (ne samo televizijskih). Kvalitet rešenja će se oceniti odgovarajućim metrikama preko ocene kvaliteta implementirane Java objektno orijentisane sprege za TV servise. Kao kriterijum za poređenje, koristiće se sledeće tri metrike: metrike za ocenu kvaliteta objektno orijentisane programske sprege, kontrolabilnost i kompletnost (sa stanovišta broja implementiranih servisa digitalne televizije).

U poglavlju 2, dat je pregled razvoja oblasti digitalne televizije u modernim uređajima potrošačke elektronike, sa osvrtom na probleme koji se postavljaju pred sistem i same okvire istraživanja.

Poglavlje 3 se bavi istraživanjem dosadašnjih dostignuća u oblasti i tu su, u cilju postavljanja okvira sistema, analizirani postojeći proizvodi na tržištu. Baza patenata sa jedne strane predstavlja veoma značajan izvor naučnih informacija, dok sa druge pruža

uvid u zaštićena rešenja koja su srodna ciljnom sistemu. Dat je i pregled savremenih naučnih dostignuća integracije TV sistemske programske podrške sa stanovišta složenosti i kvaliteta, sa ciljem da se odaberu najpogodniji postupci.

U poglavlju 4, opisane su mere koje se koriste za ocenu kvaliteta i performansi razvijene programske podrške. Pojedine mere se koriste za ocenu funkcionalnosti implementiranog sistema (npr. mera broja implementiranih servisa digitalne televizije), dok ostale obezbeđuju osnovu za merenje kvaliteta celokupnog sistema i pojedinih blokova, kao i nivo njihove kontrolabilnosti od strane viših programskih nivoa.

U poglavlju 5, dat je detaljan opis sistema, kroz opis odgovarajuće programske podrške, kao i postupaka proširenja operativnog sistema servisima digitalne televizije. Na kraju je prikazan i skup fizičkih arhitektura sposobnih da izvrše predloženo programsko rešenje u realnom vremenu, čime je potvrđena platformska nezavisnost predloženog rešenja, a samim tim i ceo sistem zaokružen u jednu celinu.

Poglavlje 6 se bavi rezultatima merenja ranije opisanim metrikama za ocenu kvaliteta predloženog rešenja. Na kraju poglavlja, izložena je uporedna analiza realizovanog rešenja sa postojećim rešenjima.

U poglavlju 7, dat je zaključak istraživanja sa posebnim osvrtom na pravce daljeg razvoja.

POGLAVLJE 2.

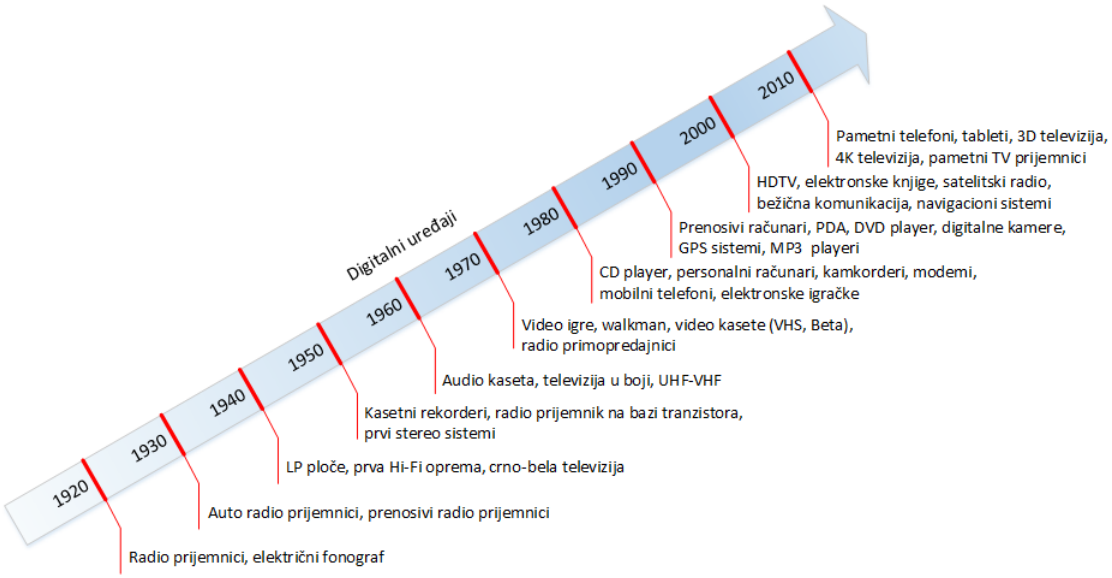
PREGLED RAZVOJA OBLASTI ISTRAŽIVANJA I POSTAVKA CILJEVA ISTRAŽIVANJA

Pojam potrošačke elektronike se prvi put javlja 1920-ih godina kao posledica masovne prodaje radio-prijemnika i elektronskih fonografa. Pojava tranzistora na početku 1950-ih, a posebno kasnija pojava integrisanih kola 1960-ih godina, dovela je do mnogih novih, i što je još važnije, poboljšanih proizvoda, kao na primer prenosivih radio-prijemnika [Papp]. Do kraja dvadesetog veka, potrošačka elektronika je postala jedan od vodećih sektora svetske ekonomije sa veoma raznovrsnom paletom proizvoda, kao što su televizori, kamkorderi, video i DVD uređaji, video igre, akustička oprema, mobilni telefoni, tableti, GPS sistemi, prenosivi i personalni računari, itd. Slika 2.1 prikazuje dominantne proizvode potrošačke elektronike u proteklim decenijama. Poseban doprinos ubrzanom razvoju oblasti uređaja potrošačke elektronike donela je upravo digitalizacija TV servisa.

Do kraja 1980-ih godina, mogućnost emitovanja potpuno digitalizovanih TV servisa korisnicima je bila nezamisliva, kako sa tehnološkog, tako i sa stanovišta ekonomske isplativosti ovakvog pristupa. Glavni razlog za ovo je velika bitska brzina neophodna za emitovanje 525, odnosno 625 digitalizovanih linija živog video toka (od 108 do 270 Mb/s nekompresovanih podataka). Drugi razlog je bio i taj što je u tom momentu akcenat stavljan na poboljšanje kvaliteta TV slike, u šta je uložena značajna

količina napora i sredstava. Ovi napori su rezultovali razvojem dva standarda za unapređen kvalitet analogne televizije:

- Televizija unapređene definicije (engl. **I**mproved **D**efinition **T**ele**V**ision, IDTV) sa brojem vertikalnih linija do 750.
- Televizija visoke definicije (engl. **H**igh **D**efinition **T**ele**V**ision, HDTV) sa brojem vertikalnih linija do 1125.



Slika 2.1 Vremenski prikaz dominantnih proizvoda iz oblasti potrošačke elektronike (izvor [Papp])

Prosta digitalizacija HDTV slika bi zahtevala bitske brzine koje su bile i do 4 puta veće nego za slike standardne rezolucije, odnosno do 1 Gb/s. Ovo je razlog zašto su mnoge inicijative (MUSE, HD-MAC) u to vreme bile definisane kao analogni sistemi sa digitalnom asistencijom i mogu se smatrati kao preteča potpuno digitalne kompresije.

Početkom 1990-ih godina, situacija na polju TV uređaja potrošačke elektronike se menja iz korena. Izuzetno brz razvoj efikasnih algoritama za kompresiju slike, koji su rezultovali prvo definicijom JPEG standardna za kompresiju nepomičnih slika, a kasnije i MPEG standardom za kompresiju pomičnih slika, pokazao je potencijal za drastično smanjenje količine podataka potrebnih za emitovanje digitalnih slika. Bitske brzine su iz domena 1 Gb/s, smanjene na raspon od 1.5 – 30 Mb/s, u zavisnosti od izabrane rezolucije i sadržaja same slike.

U isto vreme, kontinuirani razvoj integrisanih kola je dozvolio realizaciju kompleksnih integrisanih kola po pristupačnim cenama, sposobnih da izvrše dekompresiju ovako kompresovanih digitalnih slika u realnom vremenu. Za razliku od procesora opšte namene, ovakve fizičke arhitekture postižu obradu u realnom vremenu tako što obezbeđuju paralelizam na nivou fizičke arhitekture (kao što je pristup memoriji [LPRS]), odnosno posebnim tehnikama prilikom razvoja programske podrške (kao što su paralelizacija obrade [SPP], ili napredni algoritmi za digitalnu obradu signala [AADSP1]).

Cene HDTV prijemnika su i dalje bile previsoke da bi bile pristupačne većini korisnika, i to ne toliko zbog troškova elektronskih komponenti, koliko zbog cene ekrana visoke rezolucije. Ovo je rezultovalo pojavom da korisnici, i pored atraktivnosti povećane rezolucije TV uređaja visoke definicije, postaju više zainteresovani za nove servise i aplikacije koje su im nudili emiteri. Digitalno emitovanje nudi mnogo veći broj servisa koji nisu bili dostupni upotrebom analogne tehnike emitovanja. Ovi servisi uključuju emitovanje dodatnih informacija koje se emituju u transportnom toku zarad poboljšanja doživljaja gledanja televizije, ali mogu i da uključuju dodatne aplikacije koje je moguće preuzeti i pokrenuti. Ove aplikacije pružaju korisnicima nove vidove interakcije sa svojim TV uređajem, koje nisu imali prilike do tada da iskuse. Ovi vidovi interakcije mogu biti (detaljan spisak svih modernih digitalnih TV servisa se nalazi u poglavlju "Mera kompletnosti programske sprege za servise digitalne televizije"):

- Jednostavna unapređenja već ustaljenim TV servisima, kao što su višejezični prevodi ili elektronski programski vodič (engl. **E**lectronic **P**rogram **G**uide, EPG) koji pruža detaljne informacije o rasporedu emisija.
- Napredni informacioni servisi poput vesti, ili servisi vezani za emisije koje se trenutno emituju (biografije, sportske statistike).
- Nove aplikacije, kao što su elektronsko bankarstvo, interaktivne reklame itd.

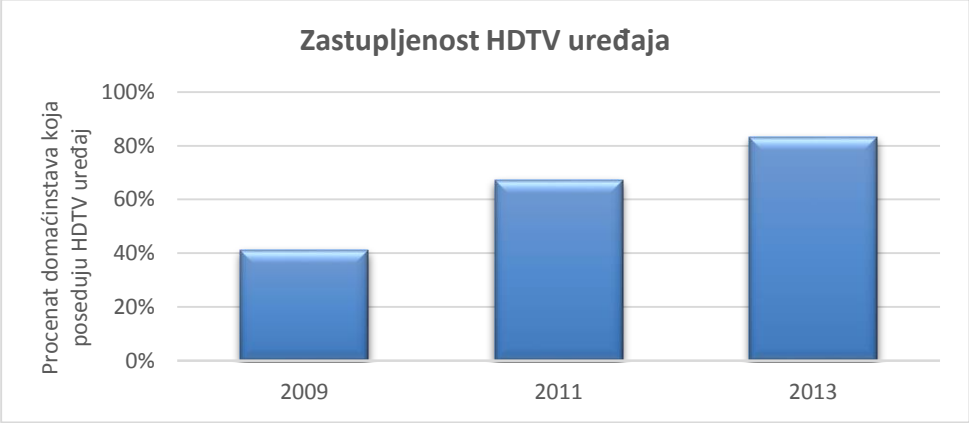
Ovo je dovelo do inicijative za definiciju sistema koji dozvoljava emitovanje digitalnih slika sa istim ili boljim kvalitetom u odnosu na trenutno stanje analognih standarda, ali sa dodatnim servisima omogućenim digitalizacijom prenosnog signala.

Jedni od prvih rezultata ove inicijative se vide u projektu "DirectTV", započetog sredinom 1994. godine u SAD.

U isto vreme, na evropskom tržištu se 1991. godine prestaje sa radom na analognoj HDTV tehnologiji (HD-MAC) i kreira se grupa za razvoj i standardizaciju digitalnih TV sistema za emitovanje. Ovo je rezultovalo DVB projektom (engl. **D**igital **V**ideo **B**roadcasting), baziranom na MPEG-2 internacionalnom standardu kompresije slike. MPEG-2 takođe ostavlja mogućnost budućeg prelaska na HDTV korišćenjem viših nivoa i profila. U periodu od 1994. do 1996. nastaju tri varijante DVB standarda, podeljene po medijumu koji se koristi za transmisiju:

- Satelitsko emitovanje (DVB-S)
- Kablovsko emitovanje (DVB-C)
- Zemaljsko emitovanje (DVB-T)

Kraj dvadesetog veka i prva decenija dvadeset prvog veka vide strm pad cena velikih ravnih ekrana visoke definicije, sa rezolucijama koje su u skladu sa HDTV zahtevima. Ovi ekrani sada postaju dostupni velikom broju korisnika, i zajedno sa novim standardima kompresije pokretne slike (kao što su MPEG-4 AVC/H.264) najzad omogućuju razvoj i popularizaciju digitalnog HDTV standarda širom sveta. Slika 2.2 prikazuje povećanje zastupljenosti HDTV prijemnika na tržištu potrošačke elektronike u poslednjih nekoliko godina.



Slika 2.2 Porast zastupljenosti HDTV uređaja u domaćinstvima (preuzeto iz [Nielsen])

Na kraju, sa sve većim unapređenjem tržišta mobilnih telefona, gde su moderni uređaji opremljeni ekranima visoke rezolucije, ali i razvojem standarda za emitovanje

digitalnih TV servisa prilagođenih mobilnim uređajima (kao što su DVB-H, T-DMB, ISDB-T, itd.), nametnulo se i pitanje razvoja i implementacije digitalnih TV servisa na ovakvim uređajima.

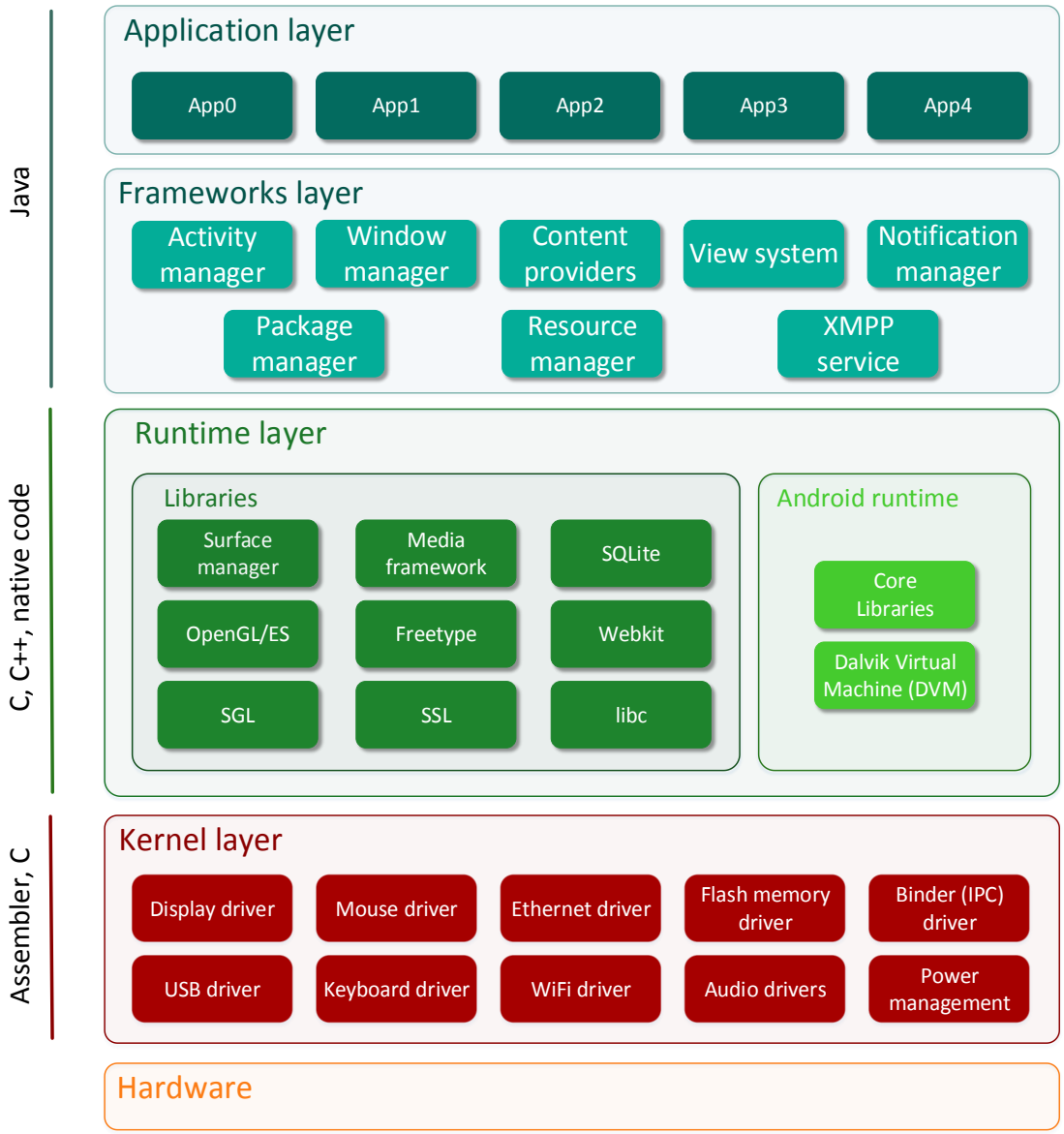
Trenutno jedan od najzastupljenijih operativnih sistema na polju tržišta mobilnih uređaja, ali i ostatka potrošačke elektronike, jeste Android ([Gargenta], [Yaghmour], [Ableson]). Ovaj operativni sistem otvorenog programskog koda, razvijan od strane kompanije Google, prvi put je najavljen 2007. godine. Prvi komercijalno raspoloživi telefon, praćen i objavljivanjem prve verzije operativnog sistema, pojavio se na tržištu krajem 2008. godine. Osnovne karakteristike ovog operativnog sistema su:

- *Aplikativno radno okruženje*: Aplikativno okruženje se koristi od strane autora aplikacija prilikom razvoja aplikacija za ovaj operativni sistem (Android aplikacije). Ovo okruženje je javno dostupno i izuzetno detaljno dokumentovano [Ableson].
- *Dalvik virtuelna mašina*: Android OS poseduje svoju implementaciju Java virtuelne mašine (ne oslanja se na standardnu Sun Java VM). Dalvik interpretira Java bajt-kod generisan od strane Java prevodioca.
- *Integrisan Internet pretraživač*: Android uključuje internet pretraživač baziran na [Webkit] kao deo standardne liste Android aplikacija. Ovaj pretraživač je na raspolaganju autorima Android aplikacija kroz standardne komponente aplikativnog radnog okruženja.
- *Optimizovana grafička sprega*: Android u sebi sadrži podršku kako za 2D tako i 3D napredne grafičke operacije.
- *SQLite*: Standardna implementacija sprege baza podataka na raspolaganju autorima Android aplikacija kroz standardne komponente aplikativnog radnog okruženja.
- *Mrežne sprege*: Android uključuje podršku za većinu bežičnih tehnologija, kao što su *Bluetooth*, *EDGE*, i *3G*.
- *Bogato razvojno okruženje*: Razvojno okruženje (engl. **S**oftware **D**evelopment **K**it, SDK) je besplatno i dostupno autorima, i uključuje emulator i alate za uklanjanje grešaka i analizu programskog koda [Android SDK].

Osim ovih karakteristika, Android operativni sistem poseduje nekoliko osobina koje ga čine posebno interesantnim kao izbor platforme za ugrađene sisteme:

- *Širok ekosistem aplikacija*: U vreme pisanja ove disertacije (januar 2014. godine) Android market broji preko 1 000 000 raspoloživih aplikacija
- *Konzistentni aplikativni API*: Sve programske sprege koje pruža aplikativno radno okruženje su namenjene tako da budu unapred-kompatibilne. Drugim rečima, sve prethodno razvijene Android aplikacije će nastaviti da funkcionišu normalno i u budućim verzijama operativnog sistema.
- *Zamenljivost komponenti*: Pošto je Android operativni sistem otvorenog tipa, većinu podrazumevanih komponenti je moguće promeniti. Primeri za ovaj princip su zamena osnovne aplikacije (engl. Home-screen launcher) ili podrazumevanog multimedijalnog radnog okruženja (Stagefright)
- *Proširivost*: Druga prednost otvorenosti Android operativnog sistema je mogućnost proširenja karakteristika platforme i dodavanja podrške za nove komponente fizičke arhitekture. Ovo je moguće korišćenjem Android HAL (engl. **H**ardware **A**bstraction **L**ayer) nivoa.

Glavni razlog integracije virtuelne mašine u ugrađenim sistemima je u vezi sa programskim jezikom koji je odabran za aplikativni razvoj. U ugrađenim sistemima, ne postoji jedan programski jezik koji je savršeno rešenje za svaki sistem, odnosno platformu. Mnogi kompleksni ugrađeni sistemi sadrže višestruke nivoe programske podrške pisanih u različitim programskim jezicima. Ovo je slučaj i sa Android operativnim sistemom, gde je nivo sistemskih uslužilaca pisan u asemblerskom i C programskom jeziku, sistemski nivo u C i C++, dok su nivoi aplikativnog radnog okruženja i samih aplikacija pisani u Java programskom jeziku. Slika 2.3 prikazuje različite slojeve programske podrške Android operativnog sistema, sa posebnim osvrtom na korišćene programske jezike prilikom realizacije tih slojeva.



Slika 2.3 Slojevi programske podrške Android operativnog sistema sa akcentom na programski jezik u kom su pisani

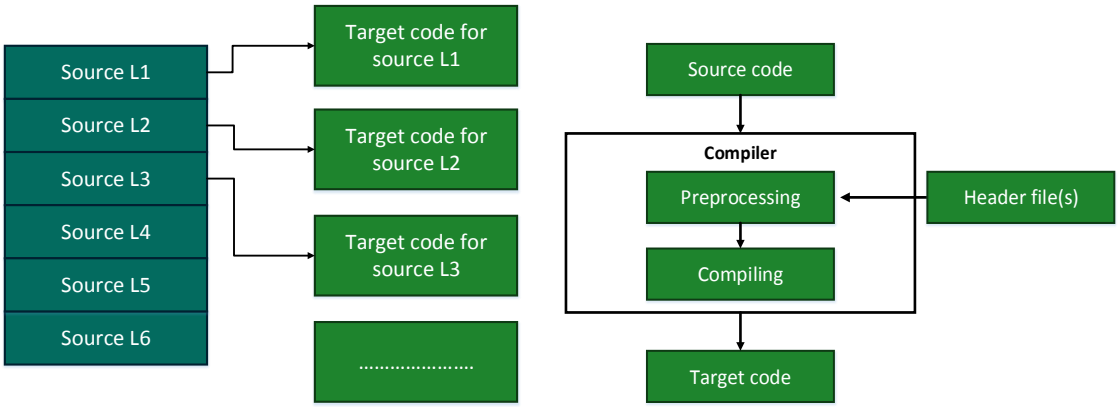
Komponente fizičke arhitekture u ugrađenim sistemima mogu direktno da izvrše mašinski programski kod, napisan mašinskim jezikom sačinjenim od nula jedinica. U vreme nastanka računarskih sistema, za programiranje se koristio isključivo mašinski jezik, što je razvoj kompleksnih aplikacija činilo gotovo nemogućim zadatkom. Da bi se unapredio proces razvoja aplikacija, programerima je omogućen pristup mašinskim instrukcijama kroz poseban skup operacija, vezanih za fizičku arhitekturu, gde je svaka operacija odgovarala jednoj ili više mašinskih instrukcija. Ovaj skup, zavistan od fizičke arhitekture, naziva se asemblerskim jezikom. Tokom vremena, drugi

programski jezici, kao što su C/C++/Java, dalje su unapredili operacijske skupove i načinili ih nezavisnim od platforme. Ovi jezici su u literaturi poznati kao jezici visokog nivoa jer su semantički još više udaljeni od mašinskog koda i nezavisni od karakteristika fizičke arhitekture. Ovo je u suprotnosti sa jezicima nižeg nivoa, kao što je asemblerski jezik, koji su zavisni od fizičke arhitekture, što ujedno znači da postoje jedinstveni operacijski skupovi za procesore sa različitim arhitekturama. Tabela 2.1 prikazuje evoluciju programskih jezika.

	Programski jezik	**Detaljniji opis**
Peta generacija	Prirodni jezici	Programski jezici slični konverzacionim jezicima, tipično korišćeni za programiranje i dizajn veštačke inteligencije (AI)
Četvrta generacija	Neproceduralni jezici veoma visokog nivoa	Objektno-orijentisani programski jezici veoma visokog nivoa, kao što su C++, C# i Java, jezici skripti (Perl i HTML), kao i jezici za pristup bazama podatka (SQL)
Treća generacija	Proceduralni jezici	Programski jezici visokog nivoa, kao što su C ili Pascal. Portabilniji od jezika prve i druge generacije
Druga generacija	Asemblerski jezik	Zavisni od fizičke arhitekture, predstavljaju mašinski kod
Prva generacija	Mašinski kod	Zavisni od fizičke arhitekture, binarne nule (0) i jedinice (1)

Tabela 2.1 Generalizovana evolucija programskih jezika (preuzeto iz [Noergaard])

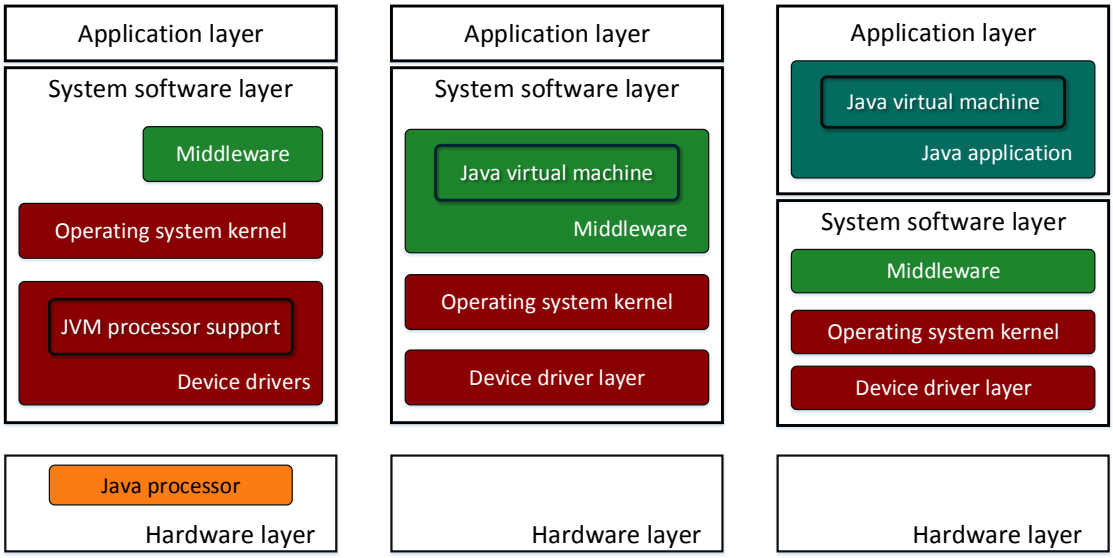
Pošto je mašinski kod jedini jezik koji fizička arhitektura može da izvršava, svi drugi jezici zahtevaju mehanizam za sintezu mašinskog koda. Ovaj mehanizam najčešće uključuje korake kao što su pred-obrada, prevođenje i interpretiranje. U zavisnosti od programskog jezika koji se izvršava, ovi mehanizmi se realizuju na razvojnom računaru (tipično neugrađena arhitektura) ili na ciljnoj platformi (ugrađeni sistem koji se razvija). Slika 2.4 prikazuje dva najčešće zastupljena mehanizma za sintezu mašinskog koda na ugrađenim platformama.



Slika 2.4 Različiti mehanizmi sinteze programskog koda: Interpretacija na ciljnoj platformi (levo) i tipično prevođenje na razvojnoj platformi (desno)

Bajt kod Java jezika je definisan tako da bude platformski nezavistan. Da bi Java bajt kod mogao da se izvršava na ugrađenim sistemima, potrebno je da na njima postoji implementacija virtuelne mašine (engl. **J**ava **V**irtual **M**achine, JVM). Postojeća rešenja na tržištu implementiraju JVM na jedan od sledeća tri načina, kao što prikazuje Slika 2.5:

- U fizičkoj arhitekturi (ARM Gazelle integrisano kolo, AJile aj100)
- Na sistemskom nivou (Ccc-J, Jeode, Jbed, Intent, KavaVM, Android)
- Na aplikativnom nivou (JVM u okviru aplikacije, Jbed, KavaVM, IBM J9)



Slika 2.5 Različite implementacije Java VM u ugrađenim sistemima ([Noergaard])

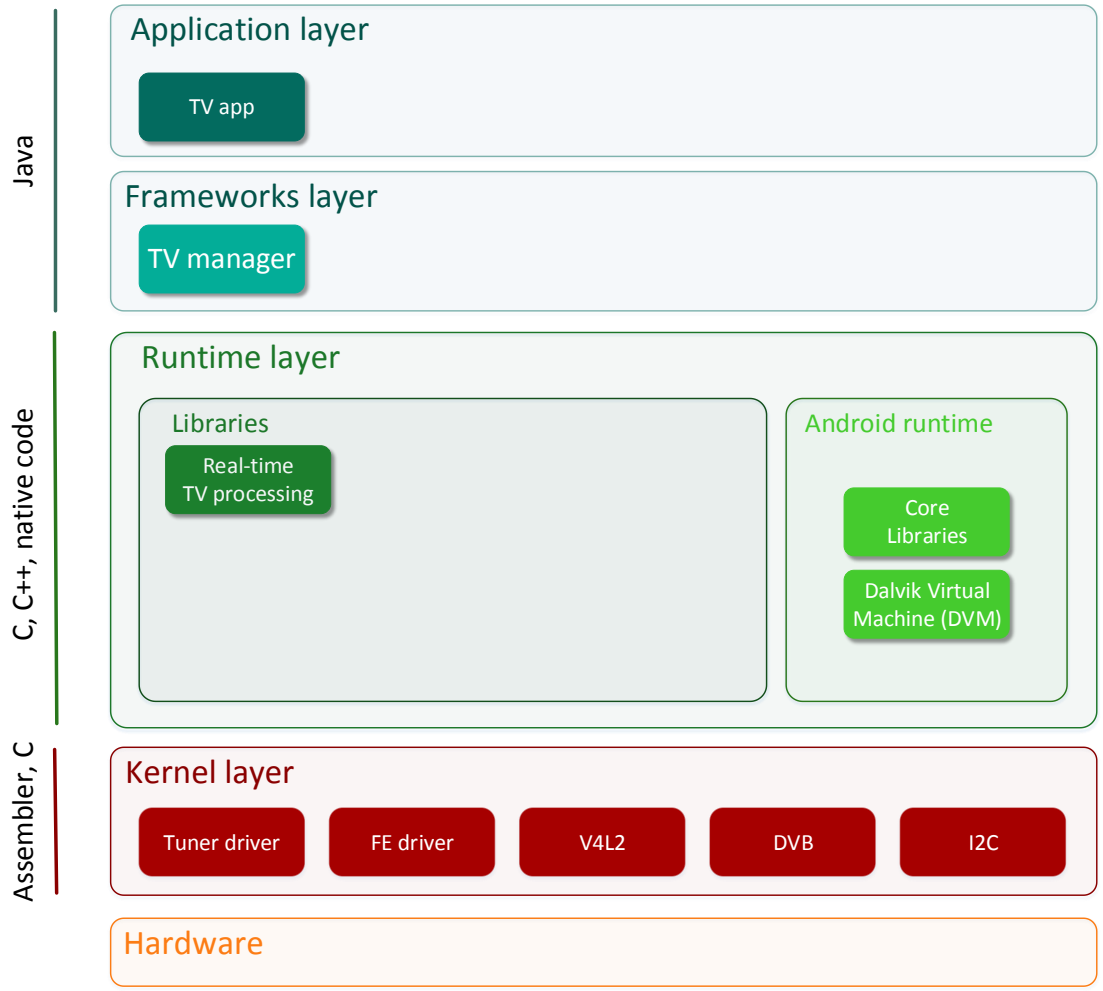
U Android operativnom sistemu, koristi se namenska implementacija virtuelne mašine pod nazivom Dalvik, koja je implementirana na sistemskom nivou, a osim interpretera sadrži i JIT prevodilac. Sa interpretacijom bajt koda u JVM, svaki bajt kod se parsira i transformiše u mašinski kod. Takođe, svi redundantni delovi koda se interpretiraju prilikom svakog izvršavanja. JIT prevodioci rešavaju ovaj problem time što program interpretiraju samo jednom, a potom prevode i skladište mašinski kod tokom izvršavanja, omogućavajući da se redundantni kod izvršava bez ponovnog interpretiranja. Iako JIT prevodioci postižu bolje performanse za redundantni kod, oni uvode dodatnu obradu u toku izvršavanja programa, koja se odnosi na pretvaranje bajt koda u mašinski. Osim dodatne obrade, JIT alati za prevođenje zahtevaju i više programske memorije za skladištenje kako Java bajt koda, tako i izvornog, prevedenog koda.

I pored svih optimizacija, Dalvik VM i Java programski jezik, isključivo, nisu pravi izbor za implementaciju programskog koda koji implementira digitalne TV servise na ugrađenoj arhitekturi. Iako je aplikativno radno okruženje Android operativnog sistema okrenuto Java programskom jeziku, implementacija servisa digitalne televizije na aplikativnom nivou nije moguća iz sledećih razloga:

- *Uslov realnog vremena*: Implementacija TV servisa mora da ispuni stroga ograničenja sa stanovišta odziva i količine podataka koji treba da se obrade u realnom vremenu. Implementacija korišćenjem Java programskog jezika, bez degradacije performansi prilikom izvršavanja, predstavlja izazov.
- *Prava pristupa*: Implementacija TV servisa se pre ili kasnije mora osloniti na sistemske uslužioce, kojima smeju da pristupe samo privilegovane, odnosno sistemske biblioteke. Regularne Android aplikacije imaju ograničena prava pristupa i nisu u mogućnosti da direktno komuniciraju sa komponentama fizičke arhitekture.
- *Životni vek*: Od TV servisa na TV uređajima se očekuje da korisniku uvek budu na raspolaganju. Uzimajući u obzir da su ugrađene TV platforme najčešće sa izuzetno ograničenim resursima, ne sme se dogoditi da usled nedostatka memorije aplikacija koja implementira TV servis

bude zatvorena (zbog standardne politike Android operativnog sistema da zatvara aplikacije ukoliko nema dovoljno memorije).

I pored ovih nedostataka, nije neuobičajeno da se Java programski jezik koristi za razvoj DTV uređaja. U literaturi se ovakav pristup [Brandao], [Lucas] oslanja putem JNI mehanizma na poseban sloj izvršnih biblioteka (engl. runtime libraries) pisanih u C/C++ jeziku. Ovaj sloj je zadužen za vremenski kritičnu obradu DTV-baziranih podataka koji dolaze iz DVB transportnog toka. Ova rešenja su bazirana na Linux OS, sa zajedničkim jezgrom na sistemskom nivou i Java VM na aplikativnom nivou zaduženom za izvršavanje aplikacija. Identičan pristup je primenljiv i na Android platforme, pri čemu bi se programska podrška za servise digitalne televizije morala raspodeliti u nekoliko programskih slojeva Android operativnog sistema. Slika 2.6 prikazuje potencijalnu raspodelu na primeru Android programske arhitekture:



Slika 2.6 Raspodela programske podrške za digitalne TV servise u Android operativnom sistemu

2.1 Postavka ciljeva istraživanja

Cilj disertacije je da predloži rešenje integracije digitalnih TV servisa u Android operativni sistem, na platformski nezavistan način, pri čemu je rešenje po složenosti uporedivo sa postojećim rešenjima na tržištu i time primenjivo u uređajima potrošačke elektronike. Digitalni TV servisi bitni za implementaciju su:

- Prikaz i pretraživanje TV servisa
- Elektronski programski vodič
- Prevod
- Teletekst
- Podsetnik
- Digitalni snimač
- Kontrolisani pristup video sadržaju
- Roditeljska kontrola
- Video na zahtev
- Kontrola ulaznih sprega

U okviru disertacije potrebno je istražiti raspoložive izvore informacija (baze patenata, izvore naučnih informacija, i postojeća tehnička rešenja), i na osnovu dobijenih rezultata postaviti ograničenja i zahteve sistema u pogledu:

- Složenosti
- Kvaliteta
- Mogućnosti

Potrebno je odabrati odgovarajuću platformu za realizaciju i potvrdu rešenja, koje treba da se uklapa u postavljena ograničenja. Disertacija će se eksperimentalno potvrditi realizacijom programske podrške na skupu ugrađenih platformi. Cilj korišćenja većeg broja različitih platformi za eksperimentalnu potvrdu je u analizi stepena platformske nezavisnosti predloženog rešenja.

Nova naučna dostignuća, koja su rezultat istraživanja, potrebno je zaštititi odgovarajućim mehanizmima zaštite intelektualne svojine.

POGLAVLJE 3.

PREGLED RELEVANTNIH INFORMACIJA

U cilju adekvatnog pozicioniranja teme istraživanja, potrebno je analizirati aktuelna dostignuća u različitim domenima. Sa stanovišta ove disertacije, smatra se da su od važnosti sledeće oblasti:

- Postojeća industrijska rešenja (uređaji) sa integrisanim digitalnim TV servisima bazirana na Androidu.
- Baze patenata.
- Naučna dostignuća u oblasti integracije digitalnih TV servisa na ugrađenim platformama.

U daljem tekstu, dat je pregled relevantnih oblasti. Nakon analize, utvrđuju se okviri istraživanja sa jasnim ciljem – proširenje Android operativnog sistema podrškom za servise digitalne TV, kao i predlog implementacije programske sprege bazirane na Java programskom jeziku, koja implementira ove servise na nivou Android aplikativnog radnog okruženja.

3.1 Postojeća industrijska rešenja sa integrisanim digitalnim TV servisima bazirana na Androidu

U okviru disertacije, istražuju se mogućnosti razvoja programske podrške za digitalne TV servise u okviru Android operativnog sistema, namenjene širokom tržištu

potrošačke elektronike. Ta činjenica nameće određene zahteve u pogledu rada u realnom vremenu, kao i konkurentnosti sa postojećim rešenjima u pogledu funkcionalnosti, kvaliteta i složenosti. Da bi se to postiglo, potrebno je analizirati postojeća industrijska rešenja po više aspekata. Ove aspekte možemo podeliti na sledeće kategorije:

- *Funkcionalnost/kompletnost*: Broj podržanih DTV servisa na uređaju. Kompletna lista standardnih DTV servisa od interesa za istraživanje prikazuje Tabela 4.1.
- *Sprega TV sveta sa Android eko-sistemom*: Stepen integracije DTV servisa sa naprednim sistemskim servisima koje pruža Android OS (pretraga podataka, sprega sa Internet pretraživačem, itd.)

Danas na tržištu postoji niz proizvoda koji nude digitalne TV servise integrisane u okviru Android operativnog sistema. Tržište možemo podeliti u dve velike grupe:

- TV/STB uređaji bazirani na originalnom Android operativnom sistemu.
- TV/STB uređaji bazirani na Google TV platformi ([GoogleTV]).

Prvu grupu predstavljaju uređaji bazirani na originalnom Android operativnom sistemu, koji je prilagođen od strane proizvođača uređaja za ciljnu platformu. Za razliku od njih, drugu grupu predstavljaju uređaji koji koriste specijalno prilagođenu verziju Android operativnog sistema za TV okruženje od strane kompanije Google (više o ovome u posebnom poglavlju "Google TV").

Slika 3.1 IPTV Android bazirani STB uređaji: ForceTech Android HD STB [ProdAndrFT] (levo) i PeerTV eTV SmartTV station [ProdAndrPeerTV] (desno)

U poslednjih nekoliko godina primetan je trend korišćenja Android operativnog sistema za ugrađene TV/STB uređaje. Rešenja možemo podeliti u dve grupe: IPTV

rešenja i hibridna rešenja. IPTV rešenja se oslanjaju na mrežnu spregu kao jedini izvor DVB transportnog toka koji se emituje putem IP protokola [Benoit]. Ovakav tip prenosa TV signala ima svoja ograničenja, među kojima je najvažnije ograničenje da se prema krajnjem korisniku u svakom momentu emituje samo jedan TV servis, i to onaj koji je korisnik u tom trenutku izabrao. Takođe, transportni tokovi koji se koriste za IPTV emitovanje imaju redukovan skup dodatnih TV servisa u sebi (najčešće nedostaju EPG informacije, SUB, ili TTX). Ovo je razlog zašto se ova rešenja ne mogu smatrati univerzalnim i integralnim rešenjima integracije TV servisa u Android operativni sistem. Druge digitalne TV karakteristike, kao što su video na zahtev, PVR, ili kontrolni pristup, podržani su u ovim rešenjima. Dva tipična primera ovakvog pristupa su uređaji: *Android HD STB* [ProdAndrFT] kompanije ForceTech i *eTV SmartTV station* [ProdAndrPeerTV] kompanije PeerTV koje prikazuje Slika 3.1.

Drugi pristup predstavljaju hibridna rešenja. Ovi uređaji, osim IPTV dela, poseduju komponentu fizičke arhitekture tjuner - koja predstavlja primarni izvor DVB transportnog toka. Primer ovakvog rešenja je *SRTAN4* [ProdAndrStrong] kompanije Strong koga prikazuje Slika 3.2. Ovaj uređaj poseduje podršku za većinu standardnih digitalnih TV servisa, izuzev uslovnog pristupa i roditeljske kontrole.

Slika 3.2 Hibridni STB uređaj baziran na Androidu: Strong, SRTAN4 [ProdAndrStrong]

Mnogi proizvodi iz domena TV/STB uređaja baziranih na Android OS pokrivaju samo segment oblasti (podskup svih standardnih DTV servisa), kao i nizak stepen integracije TV servisa u ostatak Android ekosistema. Ovo drugo se ogleda u činjenici da je TV aplikacija samo jedna od aplikacija instaliranih na sistemu, i ravnopravna u odnosu na sve ostale. Osim toga, napredne funkcije Android operativnog sistema, kao što su pretraga podataka, ne konsultuju podatke koji dolaze iz DTV sveta.

3.1.1 Google TV

Google TV predstavlja platformu, razvijenu od strane kompanije Google, koja je bazirana na Android operativnom sistemu koji proširuje određenim TV funkcionalnostima. Za razliku od Android operativnog sistema, programski kod Google TV platforme nije otvorenog tipa. Iako ova platforma, na nivou aplikativnog razvojnog okruženja, definiše programsku spregu za digitalne TV servise pisanu Java programskim jezikom, skup ovih servisa je dosta ograničen i svodi se na svega nekoliko funkcija:

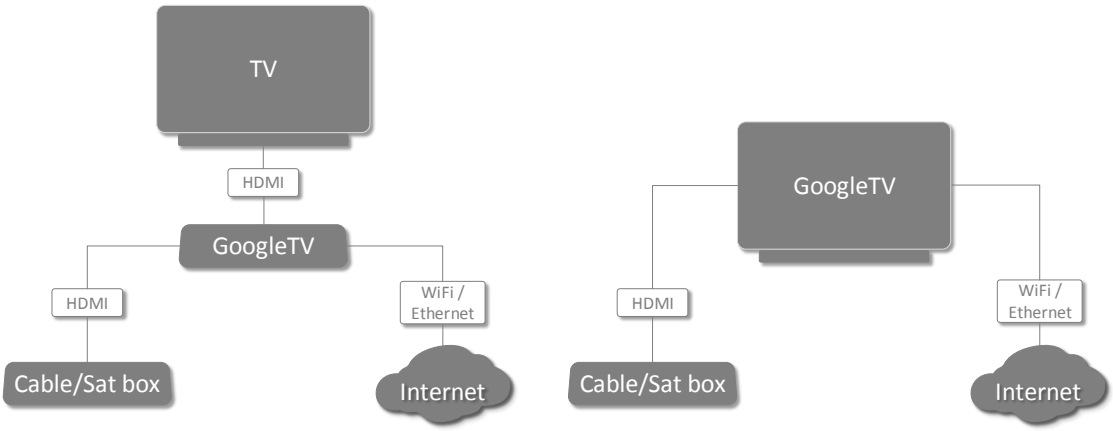
- Pretraga (engl. scan) TV servisa
- Promena kanala
- Pojačavanje zvuka TV prijemnika

Ovaj skup funkcija svakako nije dovoljan za razvoj kompleksnih TV aplikacija, koje su neophodne za moderne TV prijemnike. Sve druge standardne DTV funkcije (kao što su EPG, TTX, SUB, PVR i sl.) Google TV platforma ostavlja na nivou programske podrške specifične za uređaj na kojem se Google TV programski kod izvršava. Ova platforma nastoji da standardizuje TV funkcionalnost u okviru Android operativnog sistema, ali sa limitiranim skupom funkcija. Ovaj limitirani skup funkcija je posledica korisničkih scenarija koje Google TV platforma pokriva.

Prvi Google TV uređaji koji su se našli na tržištu, kao što su *Logitech Revue* [ProdGTVLogitech] kompanije Logitech i *Sony Internet TV Blu-ray Disc Player with Google TV* [ProdGTVSony2] kompanije Sony, zapravo i nisu prave TV/STB platforme u punom smislu te reči. Ove platforme ne poseduju komponentu fizičke arhitekture tjuner - koja ujedno i predstavlja izvor DVB transportnog toka u modernim TV uređajima. Ovi uređaji jedino poseduju ulaznu HDMI video spregu, kao jedinu komponentu fizičke arhitekture koja obezbeđuje video signal. Iz ovog razloga, ove platforme nemaju pristup dodatnim informacijama iz DVB transportnog toka, izuzev audio i video podataka, i stoga nisu ni u mogućnosti da prikažu prethodno spomenute napredne TV funkcije. Ovakav scenario upotrebe se u Google TV terminologiji zove STB posrednik (engl. buddy box).

Osim ovog pristupa, Google TV platforma podržava i integrisano rešenje, gde TV/STB prijemnik, predstavlja TV platformu u punom smislu te reči, sa svim očekivanim komponentama fizičke arhitekture. Ali, iz razloga što je aplikativno

razvojno okruženje isto za oba rešenja (i za rešenje sa STB posrednikom i za integrisani Google TV prijemnik), uređaji kao što je *Sony Internet TV with Google TV* [ProdGTVSony] kompanije Sony nude istu, limitiranu, podršku za digitalne TV servise na Java aplikativnom nivou. Slika 3.3 prikazuje ova dva korisnička scenarija.



Slika 3.3 GoogleTV korisnički scenariji: STB posrednik (levo) i integrisano rešenje (desno)

Stoga, u zavisnosti od komponenti fizičke arhitekture, Google TV uređaje možemo podeliti na integrisane i posredničke. Integrisana Google TV rešenja su implementirana u sledećim uređajima:

- *Sony Internet TV with Google TV* [ProdGTVSony], kompanije Sony
- *LG SmartTV with GoogleTV* [ProdGTVLG], kompanije LG
- *SFR TV Decoder* [ProdGTVSFR], kompanije SFR

Kod svih posredničkih rešenja, arhitektura rešenja je ista, pri čemu se uređaji razlikuju po količini usluga koje nude, a koje nisu usko vezane za TV servise. Tipični primeri ovih usluga su: video na zahtev, Internet radio, multimedijalne aplikacije i sl. Najpoznatiji uređaji ovog tipa su:

- *Asus Cube* [ProdGTVAsus], kompanije Asus
- *Hisence Pulse* [ProdGTVHisense], kompanije Hisense
- *Logitech Revue* [ProdGTVLogitech], kompanije Logitech
- *NeoTV Prime* [ProdGTVNetGear], kompanije NetGear
- *Sony Internet TV Blu-ray Disc Player with Google TV* [ProdGTVSony2], kompanije Sony

- *Internet Streaming Player Powered by Google TV* [ProdGTVSony3], kompanije Sony
- *Vizio Co-Star* [ProdGTVVisio], kompanije Visio

Slika 3.4 Google TV integrisano rešenje: TV prijemnik - [ProdGTVSony](levo) i STB – [ProdGTVSFR](desno)

Slika 3.5 Google TV uređaji za prikaz multimedijalnog sadržaja (posrednički mehanizam): Visio Co-Star (levo) i NeoTV Prime (desno)

Iako Google TV uređaji poseduju dobru integraciju DTV podataka sa naprednim funkcijama Android operativnog sistema, kao što je pretraga, ovi podaci i dalje ne dolaze iz DVB transportnog toka, već od strane namenskih servera (koje obezbeđuje i za čiju ažurnost se brine kompanija Google) putem Interneta. Ovo je ograničenje arhitekture programske podrške i kao posledicu ima potencijalnu neažurnost podataka

(podaci u transportnom toku se značajno češće osvežavaju u odnosu na podatke koji dolaze putem Interneta).

3.2 Pregled baze patenata

U današnjem poslovnom svetu tendencija svetski poznatih kompanija je da svoju intelektualnu svojinu zaštite patentima. Mnoga dostignuća i rešenja iz oblasti integracije digitalnih servisa na ugrađenim platformama, koje mogu biti i bazirane na Android OS su takođe zaštićena patentima, stoga je potrebno iscrpno istražiti i bazu dostupnih međunarodnih patenata. Osim cilja da se otklone mogući problemi u plasiranju proizvoda, patenti predstavljaju i značajan izvor informacija korisnih za istraživanje. Kao baze patenata, korišćeni su:

- Američki patentni zavod (US Patent and Trademark Office - USPTO) [USPTO].
- Evropski patentni zavod (European Patent Office - EPO) [EPO].
- Svetski patentna organizacija (World Intellectual Property Organization - WIPO) [WIPO].

Bazu patenata možemo podeliti na patente koji su usko vezani za Android operativni sistem i integraciju DTV servisa u isti, i na patente koji nisu usko vezani za Android, ali rešavaju problem integracije DTV servisa u Java baziranim sistemima i sistemima sa arhitekturom programske podrške sličnoj Android operativnom sistemu.

3.2.1 Patenti bazirani na Androidu

Patentna prijava [PatTcl] pod nazivom "*Method for integrating Android application system in television system based on Android inner core*", objavljena 07. aprila 2012, opisuje osnovnu Android implementaciju sistema digitalne televizije. Patentna prijava govori o *Binder* komunikacionom mehanizmu gde se formiraju novi slojevi komunikacije: *Binder driving* sloj, *Binder core* sloj ali i *Binder* kodovi, međutim patent ne opisuje potrebna prilagođenja Android operativnog sistema, niti tehnike za optimizaciju i prilagođenje programske sprege za digitalne TV servise.

Patentna prijava [PatLg2] objavljena od strane Lg Cns Co Ltd. pod nazivom "*Smart set-top box and operating method for smart service and digital television service using single operating system*", 2. avgusta 2012, godine predlaže metod za

upravljanje uređajem (engl. set top box) koji pruža uslugu interaktivne, „pametne“ digitalne televizije koristeći Android operativni sistem. Uređaj omogućava učitavanje aplikativne programske sprege i komunikaciju između aplikacija koje podržava upotrebom Android *Binder* mehanizma. Patent ne opisuje potrebna prilagođenja Android operativnog sistema, niti tehnike za optimizaciju i prilagođenje programske sprege za digitalne TV servise Android platformi.

Patentna prijava [PatNine] objavljena od strane Nine of Technology Co. Ltd. pod nazivom “*Android system of set top box (STB)*” predlaže različite nivoe programske podrške potrebne za razvoj DTV baziranih aplikacija u Android STB uređaju. Osim opisa funkcionalnosti ovih slojeva, patent ne daje dovoljno informacija potrebnih za rekonstrukciju rešenja, ali ukazuje na primenjeni pristup. Kao i prethodni patent, ovaj patent takođe ne opisuje potrebna prilagođenja Android operativnog sistema, niti tehnike za optimizaciju i prilagođenje programske sprege za digitalne TV servise Android platformi.

Patent [PatTianjin] objavljen od strane Tianjin, pod imenom “*Android application Loader download management system facing broadcast and TV service providers*”, iako jeste vezan za temu Android baziranih TV uređaja, ne izlaže predlog integracije celokupnog rešenja, već se koncentriše na mehanizme kojima TV operateri mogu da kontrolišu način instaliranja Android aplikacija na ciljni sistem, što čini samo jedan aspekt Android TV uređaja.

3.2.2 Drugi patenti od interesa

Uzimajući u obzir da je tema ove doktorske disertacije integracija servisa digitalne televizije u okviru Android operativnog sistema i definicija Java programske sprege za TV servise u ovom okruženju, prilikom pregleda baze patenata pretraga se može proširiti i sa generalnim principima, koji nisu usko vezani za Android operativni sistem ili principe implementacije Java programske sprege. Pošto se Android aplikativni razvojni sloj isključivo oslanja na IPC komunikaciju (*Binder* mehanizam), koja se u suštini koristi za razmenu kompleksnih Java objekata, pod optimalnom implementacijom TV servisa možemo smatrati i:

- Optimizovan model IPC komunikacije specijalizovan za ovu svrhu
- Inkrementalni pristup IPC komunikaciji

Optimizovan model IPC komunikacije, specijalizovan za digitalne TV servise, predstavlja definiciju i optimizaciju IPC prenosa zarad što efikasnije komunikacije između slojeva programske podrške u ciljnom okruženju. Uzimajući u obzir da se proces IPC razmene poruka u Android operativnom sistemu zasniva na serijalizaciji kompleksnih Java objekata na predajnoj, odnosno deserijalizaciji istih na prijemnoj strani, optimizovani model IPC komunikacije je onaj koji optimizuje količinu podataka koja se ovim putem razmenjuje za slučaj objekata digitalnih TV servisa.

Inkrementalni pristup je onaj koji uvodi zavisnost između funkcionalnosti servisa i postojećih baznih klasa na koje se dodaju, odnosno povlače dodatne potklase i podaci. Čitav metod se zasniva na optimizovanoj komunikaciji između aplikativnih slojeva (pisanih u Java programskom jeziku) i izvršnih (engl. native) slojeva pisanih u C/C++ jeziku. Osnovna ideja se ogleda u transportu osnovnog skupa podataka i objekata koji se, u zavisnosti od aktivirane funkcionalnosti servisa digitalne televizije, transportuju kroz *Binder* okruženje. Inkrementalni bazični skupovi se sastoje od skupa podataka i skupa struktura i, takođe, skupa pravila koja se šalju i optimizuju.

Patentna prijava [PatImerj] pod nazivom "*Cross-environment communication framework*", objavljena 05. aprila 2012, od strane kompanije Imerj Llc-a opisuje metod za inter-procesnu komunikaciju između servisa operativnog sistema i udaljenih objekata. Operativni sistem je operativni sistem mobilnih telefona, a optimizacija u inter-procesnoj komunikaciji nije spomenuta.

Patentna prijava [PatGoogle] pod nazivom "*Safe browser plugins using native code modules*", objavljena 13. maja 2010, opisuje kompjuterski sistem koji uvodi *plugin* mehanizam za Internet pretraživač, pri čemu sistem uključuje komunikaciju između izvršnog dela koda i samog Internet pretraživača. Patentna prijava uvodi tzv. prvi i drugi *plugin* most sprege, ali ne spominje pojam optimizacije, niti načine na koje to radi. Takođe, inkrementalno funkcionisanje mehanizma komunikacije nije spomenuto.

Patent [PatSun] pod nazivom "*Application programming interface for connecting a platform independent plug-in to a web browser*", objavljen 27. juna 2006, od strane *Sun Microsystems* govori o implementaciji aplikativne programabilne sprege za specifične *plugin* mehanizme koji su nezavisni od platforme. Takođe, inkrementalni način komunikacije, kao i optimizacija komunikacionog procesa nisu spomenuti.

Patentna prijava [PatGoogle2] pod nazivom "*Method and system for executing applications using native code modules*", objavljena 21. januara 2010, od strane Google-a govori o komunikaciji između izvršnog programskog modula i Internet aplikacije, ali ne spominje optimizacionu funkciju, niti specifičnosti inkrementalnog modela komunikacije.

Patent [PatYahoo] pod nazivom "*System for packaging native program extensions together with virtual machine applications*", objavljen 21. januara 2010, od strane Google-a govori o inter-procesnoj komunikaciji između sloja aplikacija i izvršnog programskog koda, ali ne spominje optimizacionu funkciju, niti specifičnosti inkrementalnog modela komunikacije.

Patentna prijava [PatSichuan] pod nazivom "*Interaction method of browser and hardware*", objavljena 21. januara 2010, od strane Google-a govori takođe o inter-procesnoj komunikaciji, ali ne spominje optimizacionu funkciju, niti specifičnosti inkrementalnog modela komunikacije.

Patentna prijava [PatLg] pod nazivom "*Multimedia device having operating system capable of processing multiple graphic data and method for controlling the same*", objavljena 11. jula 2012, od strane LG-a govori takođe o inter-procesnoj komunikaciji, spominje Binding, ali ne spominje optimizacionu funkciju, niti specifičnosti inkrementalnog modela komunikacije.

Patentna prijava [PatIBM] pod nazivom "*Batch dispatch of java native interface calls*", objavljena 23. decembra 2010, od strane International Business Machines Corporation takođe govori o inter-procesnoj komunikaciji, spominje Binding, ali ne spominje optimizacionu funkciju, niti specifičnosti inkrementalnog modela komunikacije.

Patent [PatHP] pod nazivom "*Method for mapping procedural C++ code to java object-oriented classes*", objavljen 05. aprila 2001, od strane Sun Microsystems-a govori o specifičnom sloju za mapiranje C++ koda u Java klase. Patent spominje i JNI sloj, međutim inkrementalni model manipulacije podataka između Java i izvršnog sloja nije spomenut.

Patent [PatSun2] pod nazivom "*Inter-process communication using different programming languages*", objavljen 22. oktobra 2001, od strane HP-a govori o specifičnom sloju za mapiranje C++ koda u Java klase. Patent spominje i JNI sloj,

međutim, inkrementalni model manipulacije podataka između Java i izvršnog sloja nije spomenut.

Patent [PatYahoo2] pod nazivom "*System for packaging native program extensions together with virtual machine applications*", objavljen 28. maja 2009, od strane Yahoo-a je veoma sličan predloženom rešenju, međutim ne daje osvrt na procedure optimizacije.

Patentna prijava [PatPhilips] pod nazivom "*Method for concurrently presenting multiple content types in a tv platform*", opisuje mehanizam istovremenog prikaza različitih sadržaja na jednoj TV platformi. Pravila uključuju kombinovanje sadržaja tipa reklama sa standardnim TV sadržajima i način njihove kompozicije prilikom prikaza krajnjem korisniku.

Patentna prijava [PatBiap] pod nazivom "*Self-contained mini-applications system and method for digital television*", slično kao i [PatPhilips], opisuje mehanizam istovremenog prikaza sadržaja na Java baziranim TV platformama, pri čemu je dodatni sadržaj u formi kartica (engl. widgets).

3.3 Naučna dostignuća u oblasti integracije digitalnih TV servisa na ugrađenim platformama

Većina modernih DTV uređaja je bazirano na *embeddedLinux* operativnom sistemu [Moon], [Miljkovic]. Mrežne i multimedijalne aplikacije nastavljaju da se razvijaju i obezbeđuju dodatnu funkcionalnost TV uređajima, koji prestaju da služe isključivo kao uređaji za gledanje TV servisa.

Jedan od prvih pokušaja integracije mrežnih i multimedijalnih servisa u DTV svet je bila pojava HbbTV [Lukac]. Ova tehnologija sakuplja korisne podatke sa Interneta, dok u isto vreme prikazuje korisniku sadržaj antenskog/kablovskog digitalnog toka. Tipične HbbTV implementacije su bazirane na proširenju Internet pretraživača. Ova proširenja obezbeđuju istovremeni prikaz sadržaja transportnog toka i podataka u CE-HTML formatu. Iako ova tehnologija jeste interaktivna i značajno unapređuje korisnikov doživljaj prilikom korišćenja DTV uređaja, ona nije standardna platforma za razvoj i korišćenje šireg skupa TV aplikacija.

Sa ciljem povećanja interaktivnosti, [Hendrawan] je predložio hibridno rešenje koje koristi namenski server koji vrši demodulaciju DTV sadržaja i potom emitovanje

istog korišćenjem IP protokola. Prijemnici ovakvih podataka su mobilni uređaji kao što su mobilni telefoni, tableti, ali i personalni računari, koji na sebi imaju instaliranu klijentsku aplikaciju. Glavni cilj ovakvih sistema je da budu korišćeni u situacijama gde jačina DVB transportnog signala nije zadovoljavajućeg kvaliteta, dok drugi mehanizmi za emitovanje podataka sa visokim protokom normalno funkcionišu (veoma brze mobilne mreže i Internet).

Neki proizvođači TV uređaja nude aplikacije koje se mogu instalirati na sam uređaj, ali su ove aplikacije prilagođene isključivo određenoj platformi. Ovo je i dovelo do pojave da se na ovakvim uređajima pređe sa tradicionalne platformski zavisne programske sprege, na platformski nezavisnu. Trenutno aktuelan trend je korišćenje Android operativnog sistema i Java programskog jezika.

3.3.1 JavaTV

U [Lucas] i [Brandao] je predloženo korišćenje [Java TV] standarda kao radnog okruženja prilikom rada sa DTV sadržajem na ugrađenim platformama. Ovaj pristup rešava problem platformski zavisne programske sprege pošto se koristi Java programski jezik za razvoj, dok se rad u realnom vremenu postiže upotrebom posebnog sloja izvršnih biblioteka koji je isključivo zadužen za parsiranje DTV podataka. Izvršni sloj je dostupan Java programskoj podršci kroz upotrebu JNI mehanizma. U ova dva rada je, takođe, predložen i skup različitih DTV aplikacija koje obezbeđuju različite servise kao što su mrežna povezanost, ali nedostaje standardizovano rešenje koje je primenljivo na različite uređaje, različitih proizvođača. Glavna mana upotrebe Java TV standarda leži i u činjenici da je on deo Java ME (Micro Edition) razvojne platforme koja je samo podskup Java SE (Standard Edition) platforme.

3.3.2 Digitalni TV prijemnici bazirani na Android OS

Iako je ova oblast relativno nova, postoje naučni radovi koji se bave problematikom integracije digitalnih TV servisa u Android operativni sistem. U [Lu] i [Kuzmanovic] prikazana su ovakva rešenja, pri čemu [Lu] predlaže sistem koji je specifičan za jednu fizičku arhitekturu. [Kuzmanovic] opisuje sistematičniji pristup, koji se bazira na sistemskoj izmeni grafičkog sistema, koji omogućuje prikazivanje DTV sadržaja na određenom delu ekrana TV prijemnika baziranom na Android OS.

Oba rešenja ne nude standardizovanu podršku za DTV servise i predstavljaju rešenja koja su prilagođena limitiranom skupu uređaja, a samim tim nisu dovoljno univerzalna i kompletna.

Sa druge strane, u [Vidakovic] i [Vidakovic2], prikazan je drugačiji pristup DTV integraciji u Android OS. Ovaj pristup ne zahteva izmene originalnog operativnog sistema i omogućuje rad na širokom skupu uređaja (ugrađeni uređaji, ali i personalni računari). Radovi predlažu i Java API, ali i API za izvršni sloj koji se može koristiti kao osnova za standardizaciju DTV aplikacija pisanih Java programskim jezikom. Takođe, ovi radovi nastoje da postave standarde u načinu na koji se servisi digitalne televizije integrišu u ovaj operativni sistem. Osim ova dva rada, u [Kuzmanovic2] i [Lukic], predloženo rešenje iz [Vidakovic2] je dodatno prošireno sa podrškom za HbbTV i pretragu DTV baziranih podataka. Ova doktorska disertacija se oslanja upravo na radove iz [Vidakovic2] i [Lukic] i predstavlja njihovo detaljno izlaganje.

POGLAVLJE 4.

PREGLED MERA ZA OCENU KVALITETA PROGRAMSKE SPREGE ZA DIGITALNE SERVISE

U fokusu istraživanja, nalazi se razvoj programske podrške koja predstavlja proširenje Android operativnog sistema servisima digitalne televizije i koja treba da postigne izvršavanje u realnom vremenu na odgovarajućoj fizičkoj arhitekturi.

Uzimajući u obzir karakteristike rešenja koje se razvija, ocena kvaliteta će se obaviti u nekoliko koraka. Prvo, uzimajući u obzir da je okruženje za aplikativni razvoj u Android OS bazirano na Java programskom jeziku, predloženo programsko rešenje će biti ocenjeno metrikama za analizu OO (Objektno-orijentisanog) programskog dizajna. Ove metrike imaju za cilj da ukažu na greške prilikom osmišljavanja arhitekture programske podrške ali i potencijalne greške koje se mogu ispoljiti prilikom održavanja i ponovnog iskorišćenja predloženog programskog rešenja.

Dalje, uzimajući u obzir trendove na polju potrošačke elektronike u modernim TV/STB sistemima, predloženo rešenje će biti analizirano sa aspekta kompletnosti i kontrolabilnosti. Kompletnost će biti posmatrana kao broj podržanih modernih servisa digitalne televizije, dok je cilj ocene kontrolabilnosti da ukaže na sposobnost rešenja da isprati dinamiku i protok velike količine informacija prisutne u modernim TV platformama.

4.1 Mere za ocenu kvaliteta objektno orijentisane programske sprege

Upotreba metrika da bi se kontrolisao, predvideo, i unapredio kvalitet proizvoda koji je baziran na programskoj sprezi (API) dobija na značaju i popularnosti. Postoji veliki izbor različitih tipova metrika koje se mogu koristiti u toku životnog veka projekta, ali će akcenat, s obzirom na domen predložene programske podrške koja je opisana u ovoj disertaciji, biti dat objektno orijentisanim (engl. **O**bject-**O**riented, OO) metrikama za merenje kvaliteta programske podrške. Pošto sam odabir OO paradigme za razvoj programske podrške nije dovoljan da sam po sebi garantuje kvalitet, cilj OO metrika je da analiziraju sistem u cilju postizanja što boljeg kvaliteta.

Neke tradicionalne metrike, ili njihove izmene, mogu biti korisne u okviru OO paradigme, posebno na nivou metoda klasa. Da bi se kvantifikovale OO karakteristike, kao što su: enkapsulacija, skrivanje informacija, nasleđivanje, polimorfizam i propuštanje poruka, mnogi autori su predložili različite, najčešće, skupove metrika. U ovoj disertaciji, fokus je na metrike koje su definisali autori Abreu [Abreu1], Chidamber i Kemerer [Chidamber1], ali i na podskup metrika koje su definisali Lorenz i Kidd [Lorenz]. Metrike će biti objašnjene i svrstane spram klasifikacije koju je definisao autor Archer [Archer], koja prati i pokriva sve aspekte i granularnost OO baziranih sistema. Za svaku metriku, biće naglašene njene osnovne karakteristike, kako se izračunava, njena prilagođenost i korisnost. Granični pragovi za neke od metrika su empirijski utvrđeni [Lorenz], ali i dalje ne postoji usaglašena skala za svaku od metrika i najčešće se pragovi određuju spram programske podrške, odnosno problema koji se rešava.

Reprezentativni skup metrika koji je korišćen u ovoj disertaciji je opisan u skladu sa klasifikacijom definisanom u literaturi [Archer] i na način koji oslikava sve moguće specifičnosti i granularnost OO paradigme.

4.1.1 Metrike na sistemskom nivou

Metrike na sistemskom nivou se mogu izvesti iz metrika na nivou klasa sa odgovarajućom statistikom, čineći tako relativne mere koje identifikuju sisteme programske podrške koji odstupaju od norme. MOOD (engl. **M**etrics for **O**bject

Oriented **D**esign) skup metrika [Abreu1] analizira programsku podršku na sistemskom nivou, prateći OO paradigmu i mehanizme kao što su:

- enkapsulacija (MHF i AHF)
- nasleđivanje (MIF i AIF)
- polimorfizam (POF)
- propuštanje poruka (COF)

U radovima [Abreu1], [Abreu2], [Abreu3], [Abreu4], [Harrison2] i [Al-Ja'afer1] predloženi su minimalni i maksimalni dozvoljeni pragovi za svaku od prethodno navedenih metrika. Ukoliko programska podrška koja se analizira zadovoljava predloženi raspon pragova, može se smatrati da ispunjava uslove kvaliteta sa stanovišta objektno orijentisane paradigme, na sistemskom nivou. Predloženi pragovi za svaku od metrika će biti navedeni prilikom njihovog detaljnijeg opisa u narednim pasusima.

4.1.1.1 Faktor skrivanja metoda

Faktor skrivanja metoda (engl. **M**ethod **H**iding **F**actor, MHF) je definisan kao odnos sume stepena nevidljivosti svih metoda definisanih u svim klasama i sume broja metoda definisanih u sistemu koji se analizira. Stepen nevidljivosti neke metode je procenat od ukupnog broja klasa iz kojih ta metoda nije vidljiva. Drugim rečima, MHF je odnos skrivenih metoda (bilo zaštićenih ili privatnih) i ukupnog broja metoda:

$$MHF = \frac{\sum_{i=0}^{TC}\sum_{m=1}^{M_d(C_i)}\left(1 - V(M_{mi})\right)}{\sum_{i=1}^{TC} M_d(C_i)}$$

pri čemu je $M_d(C_i)$ broj metoda deklarisanih u klasi C_i a:

$$V(M_{mi}) = \frac{\sum_{j=1}^{TC} is\_visible\left(M_{mi}, C_j\right)}{TC - 1}$$

i:

$$is\_visible\left(M_{mi}, C_j\right) = \begin{cases} 1, & akko\ j \neq i\ \wedge\ C_j može\ da\ zove\ M_{mi} \\ 0, & za\ ostale\ vrednosti \end{cases}$$

za sve klase $C_1, \ldots C_n$, metoda se računa kao 0 ukoliko može da se koristi u drugoj klasi, a kao 1 ukoliko ne može. *TC* predstavlja ukupan broj klasa u programskoj podršci koja se analizira.

Namena MHF je da meri enkapsulaciju, odnosno relativni odnos skrivenih informacija. Utvrđeno je u [Abreu1] da se sa porastom MHF, količina grešaka i napor

da se iste isprave smanjuje. Sa stanovišta opsega dozvoljenih vrednosti za ovu metriku, u literaturi je za minimalnu dozvoljenu vrednost predložen prag od 12.7%, dok je za maksimalnu dozvoljenu vrednost postavljen prag od 21.8%.

4.1.1.2 Faktor skrivanja atributa

Faktor skrivanja atributa (engl. **A**ttribute **H**iding **F**actor, AHF) je definisan kao odnos sume stepena nevidljivosti svih atributa definisanih u svim klasama i sume broja atributa definisanih u sistemu koji se analizira. Drugim rečima, AHF je odnos skrivenih atributa (bilo zaštićenih ili privatnih) i ukupnog broja atributa:

$$AHF = \frac{\sum_{i=0}^{TC} \sum_{m=1}^{A_d(C_i)} \left(1 - V(A_{mi})\right)}{\sum_{i=1}^{TC} A_d(C_i)}$$

pri čemu je $M_d(C_i)$ broj metoda deklarisanih u klasi C_i a:

$$V(A_{mi}) = \frac{\sum_{j=1}^{TC} is\_visible\left(A_{mi}, C_j\right)}{TC - 1}$$

i:

$$is\_visible\left(A_{mi}, C_j\right) = \begin{cases} 1, akko\ j \neq i\ \wedge\ C_j može\ da\ koristi\ A_{mi} \\ 0, \qquad\qquad za\ ostale\ vrednosti \end{cases}$$

za sve klase $C_1, \ldots C_n$, atribut se računa kao 0 ukoliko može da se koristi u drugoj klasi, a kao 1 ukoliko ne može. *TC* predstavlja ukupan broj klasa u programskoj podršci koja se analizira.

Namena AHF je, takođe, da meri enkapsulaciju, odnosno relativni odnos skrivenih informacija. Idealno, ova metrika bi uvek trebalo da ima vrednost 100%. Programska podrška bi, po pravilu, trebalo da krije skoro sve instance podataka. Korišćenje javnih atributa se ne ohrabruje pošto narušava OO enkapsulaciju kroz otkrivanje implementacije objekta. U retkim slučajevima, javni atributi se mogu opravdati potrebom za visokim performansama sistema, pa se izbegava korišćenje metoda za dobavljanje i postavljanje atributa klase (engl. getters and setters).

Sa stanovišta opsega dozvoljenih vrednosti za ovu metriku, u literaturi je za minimalnu dozvoljenu vrednost predložen prag od 75.2%, dok je za maksimalnu dozvoljenu vrednost postavljen prag od 100%.

4.1.1.3 Faktor nasleđivanja metoda

Faktor nasleđivanja metoda (engl. **M**ethod **I**nheritance **F**actor, MIF) definiše se kao odnos sume nasleđenih metoda u svim klasama programske podrške koja se analizira i ukupnog broja raspoloživih metoda (lokalno definisanih i nasleđenih) za sve klase. MIF direktno meri broj nasleđenih metoda kao podskup ukupnog broja metoda.

$$MIF = \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)}$$

gde je:

$$M_a(C_i) = M_d(C_i) + M_i(C_i)$$

i:

$M_d(C_i)$: broj metoda deklarisanih u klasi

$M_a(C_i)$: broj metoda koje se mogu pozvati iz klase C_i

$M_i(C_i)$: broj nasleđenih metoda (i ne preklopljenih)

TC: ukupan broj klasa u programskoj podršci koja se analizira

U [Abreu1], predložen je MIF kao mera nasleđivanja i posledično kao mera ponovne iskoristivosti programske podrške. Upotreba nasleđivanja se može gledati kao tip ustupka i biće diskutovana u više detalja u okviru poglavlja "Metrike na nivou nasleđivanja". Neke od empirijskih studija koje analiziraju faktor nasleđivanja su [Daly], [Harrison1] i [Harrison2].

Sa stanovišta opsega dozvoljenih vrednosti za ovu metriku, u literaturi je za minimalnu dozvoljenu vrednost predložen prag od 66.4%, dok je za maksimalnu dozvoljenu vrednost postavljen prag od 78.5%.

4.1.1.4 Faktor nasleđivanja atributa

Faktor nasleđivanja atributa (engl. **A**ttribute **I**nheritance **F**actor, AIF) definiše se kao odnos sume nasleđenih atributa u svim klasama programske podrške koja se analizira i ukupnog broja raspoloživih atributa (lokalno definisanih i nasleđenih) za sve klase. Metrika je definisana na identičan način kao i MIF. AIF direktno meri broj nasleđenih atributa kao podskup ukupnog broja atributa.

$$AIF = \frac{\sum_{i=1}^{TC} A_i(C_i)}{\sum_{i=1}^{TC} A_a(C_i)}$$

gde je:

$$A_a(C_i) = A_d(C_i) + A_i(C_i)$$

i:

$A_d(C_i)$: broj atributa deklarisanih u klasi

$A_a(C_i)$: broj atributa koji se mogu pozvati iz klase C_i

$A_i(C_i)$: broj nasleđenih atributa

TC: ukupan broj klasa u programskoj podršci koja se analizira

AIF, kao i MIF, meri nivo nasleđivanja u sistemu i posledično meri ponovnu iskoristivost programske podrške. Preveliki stepen ponovnog iskorišćenja kroz nasleđivanje može ponekad voditi ka programskoj podršci koja je teža za razumevanje, održavanje i testiranje.

Sa stanovišta opsega dozvoljenih vrednosti za ovu metriku, u literaturi je za minimalnu dozvoljenu vrednost predložen prag od 52.7%, dok je za maksimalnu dozvoljenu vrednost postavljen prag od 66.3%.

4.1.1.5 Faktor polimorfizma

Faktor polimorfizma (engl. **PO**lymorphism **F**actor, POF) definiše se kao odnos stvarnog broja različitih polimorfnih scenarija za klasu C_i, i teorijskog maksimuma polimorfnih situacija za istu tu klasu C_i. POF je broj metoda koje redefinišu nasleđene metode, podeljen maksimalnim brojem mogućih različitih polimorfnih situacija.

$$POF = \frac{\sum_{i=1}^{TC} M_o(C_i)}{\sum_{i=1}^{TC} [M_n(C_i) \times DC(C_i)]}$$

gde je:

$$M_d(C_i) = M_n(C_i) + M_o(C_i)$$

i:

$M_n(C_i)$: broj novih metoda klase C_i

$M_o(C_i)$: broj nasleđenih metoda klase C_i

$DC(C_i)$: broj klasa koje nasleđuju klasu C_i

TC: ukupan broj klasa u programskoj podršci koja se analizira

POF meri potencijal za polimorfizam koji određena programska podrška ima. Pošto polimorfizam proizilazi iz nasleđivanja, ova mera indirektno analizira i stepen dinamičkih veza koje postoje u sistemu. U ekstremnim slučajevima, može se desiti da metrika nema definisanu vrednost (za sisteme programske podrške koji u sebi nemaju nasleđivanje).

Sa stanovišta opsega dozvoljenih vrednosti za ovu metriku, u literaturi je za minimalnu dozvoljenu vrednost predložen prag od 2.7%, dok je za maksimalnu dozvoljenu vrednost postavljen prag od 9.6%.

4.1.1.6 Faktor sprege

Faktor sprege (engl. **CO**upling **F**actor, COF) definiše se kao odnos maksimalnog broja mogućih sprega između klasa i stvarnog broja sprega koje se ne mogu pripisati nasleđivanju. Drugim rečima, ova metrika broji količinu komunikacije između klasa u sistemu.

$$COF = \frac{\sum_{i=1}^{TC}\left[\sum_{j=1}^{TC} is\_client(C_i, C_j)\right]}{TC^2 - TC}$$

gde je:

$$is\_client(C_c, C_s) = \begin{cases} 1, akko\ (C_c \Rightarrow C_s) \wedge\ C_c \neq C_s \\ 0, \quad za\ ostale\ vrednosti \end{cases}$$

a:

TC: ukupan broj klasa u programskoj podršci koja se analizira

U prethodnoj jednačini, $C_c \Rightarrow C_s$ predstavlja odnos potrošač-proizvođač, što u domenu OO programske sprege znači da jedna potrošačka klasa C_c sadrži najmanje jednu referencu (nenasleđenu) na metodu proizvođačke klase C_s. Sprega između klasa se može posmatrati kao mera povećanja kompleksnosti, smanjenja kako enkapsulacije, tako i potencijala ponovne iskoristivosti i na kraju razumljivosti programske sprege. U [Abreu1], utvrđena je pozitivna korelacija između povećanja COF mere i količine grešaka u programskoj podršci koja se analizira.

Sa stanovišta opsega dozvoljenih vrednosti za ovu metriku, u literaturi je za minimalnu dozvoljenu vrednost predložen prag od 0%, dok je za maksimalnu dozvoljenu vrednost postavljen prag od 11.2%.

4.1.2 Metrike na nivou sprege klasa

Sprega klasa predstavlja mehanizam gde se metode ili atributi definisani u jednoj klasi, koriste direktno u drugoj klasi. Pomenute klase tada, ovim vidom sprege formiraju odnos sistem/podsistem i predstavljaju ranu indikaciju kompleksnosti dizajna programske podrške koja se analizira. Reprezentativna metrika koja kvantifikuje ovu pojavu je opisana u [Chidamber1].

4.1.2.1 Sprega između objekata klasa

Sprega između objekata klasa (engl. **C**oupling **B**etween **O**bjects, CBO) za određenu klasu se definiše kao broj drugih klasa sa kojima je ova klasa spregnuta. Sprega između klasa se događa kada jedna klasa koristi metode ili atribute druge klase. CBO se meri brojanjem različitih zavisnosti jedne klase od drugih klasa, koje nisu vezane za nasleđivanje. U [Chidamber1], je predloženo da se ova mera koristi kao indikator napora potrebnog za održavanje i testiranje programske podrške.

Što je veća sprega između klasa, teže je ponovo iskoristiti datu klasu prilikom održavanja ili unapređenja programske podrške. Klase bi trebalo da su što nezavisnije jedne od drugih, da bi mehanizam ponovnog iskorišćenja mogao da funkcioniše na optimalnom nivou. U praksi, klase uvek imaju međuzavisnost, ali ukoliko je ova međuzavisnost prevelika, razumevanje dizajna programske podrške postaje toliko komplikovano da se ponovno iskorišćenje klase češće zamenjuje ponovnim pisanjem klasa.

Klase koje imaju visoku vrednost za CBO su teže za održavanje i moraju se mnogo rigoroznije testirati. Prisutnost jakih sprega u sistemu komplikuje isti, pošto, zbog prisustva kompleksnih sprega, dizajn postaje teži za razumevanje, ali i održavanje. Sa druge strane, sistem koji ne poseduje ovu pojavu, najčešće ima nižu kompleksnost, veću modularnost i više podržava paradigmu enkapsulacije.

U literaturi ne postoji jasno definisani pragovi dozvoljenih vrednosti za ovu metriku. Iz tog razloga, rezultati dobijeni merenjem ovom metrikom se mogu posmatrati isključivo kao komparativna analiza ocene kvaliteta pojedinih klasa u sistemu.

4.1.3 Metrike na nivou nasleđivanja

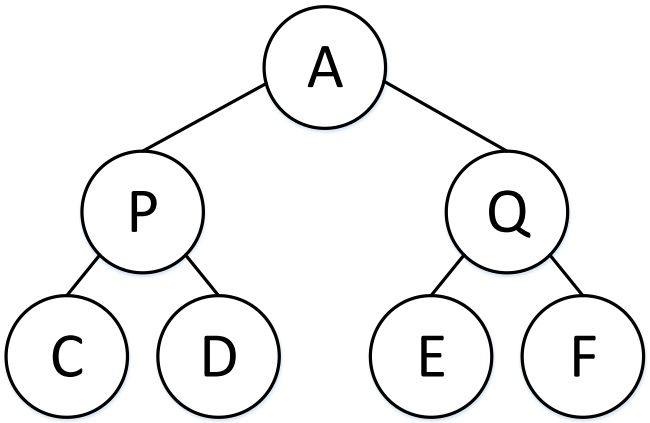
Nasleđivanje predstavlja jedan od osnovnih principa OO dizajna. Ovaj odnos se može posmatrati kao svojevrstan ustupak. Pored prednosti kao što su ponovno iskorišćenja klase, koje se odnosi na dodavanje dodatnih karakteristika određenoj klasi, bez izmena same klase, nasleđivanje, ukoliko se intenzivno koristi može dovesti do pojave da programski kod postane težak za razumevanje i održavanje. Ovaj problem je u literaturi poznat pod imenom “yo-yo” problem [Gamma]. Posebno se javlja problem ukoliko se javi potreba za promenom sprega definisanih u osnovnoj klasi.

Takođe, primećen je trend smanjenja upotrebe nasleđivanja, izmeren pomoću metrika opisanih u nastavku (DIT, NOC, [Chidamber1]) na primerima programske podrške koja se koristi u postojećim korporativnim sistemima [Cartwright], [Chidamber2].

Kao alternativa nasleđivanju, tehnika kompozicije se može razmatrati jer pruža pristup koji dozvoljava lakšu i jednostavniju promenu programskog koda, a i dalje sa ciljem održavanja stepena ponovne iskoristivosti. Sa druge strane, višestruko nasleđivanje se ne preporučuje zbog potencijalne kolizije imena u programskim jezicima koji dozvoljavaju višestruke osnovne klase, kao što je C++. Uzimajući u obzir da je cilj ove disertacije razvoj programske sprege korišćenjem Java programskog jezika, problem višestrukog nasleđivanja se neće razmatrati.

4.1.3.1 Dubina stabla nasleđivanja

Dubina stabla nasleđivanja [Chidamber1] (engl. **D**epth of **I**nheritance **T**ree, DIT) meri maksimalni nivo hijerarhije nasleđivanja za jednu klasu. Koren stabla nasleđivanja ne nasleđuje nijednu klasu i stoga ima vrednost nula za DIT. Sve ostale klase računaju svoju DIT vrednost direktnim sabiranjem broja nivoa nasleđivanja od korena stabla do njih samih.



Slika 4.1 Primer stabla hijerarhije nasleđivanja

Što je klasa dublje u hijerarhiji nasleđivanja, veći je i potencijalni broj metoda koje nasleđuje, čineći klasu kompleksnijom i težom za predviđanje njenog ponašanja. Takođe, dublja stabla nasleđivanja konstruišu i veću kompleksnost dizajna programske

sprege uzimajući u obzir broj metoda koje su u igri. Sa druge strane, što je klasa dublje u hijerarhiji, veća je potencijalna ponovna iskoristivost nasleđenih metoda.

Sa stanovišta sistema programske podrške, velike vrednosti za DIT predstavljaju signal da se u sistemu nalaze kompleksni objekti, potencijalno teški za testiranje. Sa druge strane, male vrednosti DIT metrike upućuju na to da sistem sadrži programski kod koji ne iskorišćava na pravi način prednost mehanizma nasleđivanja u OO dizajnu. U radu [Lorenz], predložen je prag za DIT metriku u vrednosti 6 za individualne klase, dok je u drugim radovima [Cartwright] istražen odnos DIT metrike i broja problema koje su prijavili krajnji korisnici sistema.

Nedostaci DIT metrike se mogu analizirati iz nekoliko različitih uglova. U programskim jezicima kao što su Java, koji je od interesa u ovoj disertaciji, svaka klasa nasleđuje makar *Object* klasu, što dodaje vrednost jedan za DIT za svaku klasu u sistemu. Osim ovoga, veći problem DIT metrike je njena definicija, ali i validnost upotrebe prilikom merenja stepena ponovne iskoristivosti u sistemu, jer se u praksi može desiti da klase sa visokom vrednosti za DIT, koriste jednak ili manji broj metoda od plitkih a širokih hijerarhija nasleđivanja. Kao podrška DIT metrici, može se posmatrati metrika NMI (engl. **N**umber of **M**ethods **I**nherited) – broj nasleđenih metoda, opisana u [Lorenz].

4.1.3.2 Broj izvedenih klasa

Broj izvedenih klasa [Chidamber1] (engl. **N**umber **O**f **C**hildren, NOC) predstavlja broj neposrednih izvedenih klasa jedne klase. Drugim rečima, ova metrika broji koliko izvedenih klasa će da nasledi metode osnovne klase. Interpretacija metrike se može posmatrati iz sledećih uglova:

- Veći broj klasa koje nasleđuju jednu klasu, veća ponovna iskoristivost, pošto je nasleđivanje forma ponovne iskoristivosti.
- Veći broj klasa koje nasleđuju jednu klasu, veća je verovatnoća da osnovna klasa nije dobro apstrahovana. Takođe, može se razmatrati i pogrešna upotreba izvedenih klasa.
- Broj izvedenih klasa takođe daje ideju koliki je potencijalni uticaj same klase na celokupan dizajn sistema programske podrške. Veća vrednost nagoveštava veću potrebu za testiranjem metoda date klase.

Indeks specijalizacije klase [Lorenz] (engl. **S**pecialisation **I**nde**X** per Class, SIX) meri stepen koliko izvedene klase preklapaju ponašanje osnovne klase.

$$SIX = \frac{NOC * DIT}{TM}$$

gde je:

NOC: Broj preklopljenih metoda klase

DIT: Prethodno opisana metrika, dubina stabla nasleđivanja

TM: Ukupan broj metoda

SIX metrika signalizira da se metode u sistemu preklapaju u tolikoj količini da se dovodi u pitanje apstrakcija i njena ispravnost, s obzirom na to da se značajna količina ponašanja mora izmeniti. Izvedene klase bi trebalo da proširuju ponašanje klase koju nasleđuju novim metodama, a ne da zamenjuju ili brišu ponašanje kroz preklapanje. Ideja specijalizacije klasa je pravljenje klasa koje predstavljaju nadskup, i kao karakteristiku imaju mali broj preklopljenih metoda, smanjen broj novih metoda i mali broj ili izostanak brisanja metoda. U [Lorenz], predložena je vrednost od 15% za SIX kao prag za identifikovanje klasa za koje je potrebna korektivna akcija, odnosno izvedena klasa nema puno dodira sa osnovnom klasom.

4.1.4 Metrike na nivou klasa

Ova grupa metrika identifikuje karakteristike u okviru jedne klase, naglašavajući različite aspekte apstrakcije klase i mogućih akcija koje se mogu primeniti ukoliko se dizajn programske podrške ili implementacija pokažu kao pogrešne.

4.1.4.1 Odziv klase

Odziv klase [Chidamber1] (engl. **R**espone **F**or a **C**lass, RFC) predstavlja broj svih metoda koje se mogu prozvati kao odgovor na poruku objektu klase ili putem metode u klasi. Ovo uključuje sve metode kojima se može pristupiti u hijerarhiji klasa. RFC broji moguće pozive ka drugim klasama iz određene klase:

$$RFC = |RS|$$

pri čemu je *RS* odzivni skup za klasu i može se definisati kao:

$$RS = \{M\} \cup_i \{R_i\}$$

gde je:

$\{R_i\}$: skup metoda zvanih iz metode *i*

$\{M\}$: skup svih metoda u klasi

RFC je mera kompleksnosti klase koja se odražava kroz broj metoda i količinu komunikacije između klasa. RFC se može posmatrati i kao indikator potrebne količine testiranja i ispravljanja grešaka. Što je veća RFC vrednost, to je veća kompleksnost zbog velike količine metoda koje se mogu prozvati, a samim tim i količina vremena potrebna za testiranje.

U literaturi ne postoji jasno definisani pragovi dozvoljenih vrednosti za ovu metriku. Iz tog razloga, rezultati dobijeni merenjem ovom metrikom se mogu posmatrati isključivo kao komparativna analiza ocene kvaliteta pojedinih klasa u sistemu.

4.1.4.2 Ponderisane metode za klasu

Ponderisane metode za klasu [Chidamber1] (engl. **W**eighted **M**ethods per **C**lass, WMC) mere kompleksnost individualnih klasa. Ukoliko se smatra da je svaka metoda u klasi jednako kompleksna, onda WMC predstavlja broj definisanih metoda u klasi.

$$WMC = \sum_{i+1}^{n} C_i$$

gde klasa C_1 ima $M_1, ..., M_n$ metoda sa $c_1, ..., c_n$.

Pošto WMC meri kompleksnost klase, može se posmatrati kao indikator napora potrebnog za razvoj i održavanje klase. Klase koje imaju veliki broj metoda imaju uticaj na klase koje ih nasleđuju, jer od njih nasleđuju sve metode. Osim toga, ovakve klase su specifične samo za određene aplikacije, pri čemu se smanjuje potencijal za ponovno iskorišćenje. U [Lorenz], predložen je prag od 20 do 40 metoda, u zavisnosti od toga da li se klase koriste za razvoj grafičke korisničke sprege. U drugim radovima [Harrison1], ističe se mana ove metrike, a to je da je vreme potrebno za razvoj klase sa jednom velikom kompleksnom metodom uporedivo sa vremenom potrebnim za razvoj klase sa mnogo jednostavnih metoda. Stoga se ova metrika može najbolje koristiti kao mera veličine klase.

4.1.4.3 Nedostatak kohezije između metoda klase

Nedostatak kohezije između metoda klase [Chidamber1] (engl. **L**ack of **CO**sehion in **M**ethods, LCOM) meri do koje mere metode referenciraju podatke od klase. Ukoliko posmatramo klasu C_1 koja ima n metoda $M_1, M_2, ..., M_n$, tada možemo

definisati $\{I_j\}$ kao skup promenljivih klase koje se koriste u metodi M_i. Tada postoji n skupova $\{I_1\}, \ldots, \{I_n\}$. Ukoliko, potom, definišemo:

$$P = \{(I_i, I_j) | I_i \cap I_j = \emptyset\}$$

$$Q = \{(I_i, I_j) | I_i \cap I_j \neq \emptyset\}$$

tada je i, ukoliko su svih n skupova $\{I_1\}, \ldots, \{I_n\}$ prazni, $P=\emptyset$.

$$LCOM = \begin{cases} |P| - |Q|, ako\ je\ |P| > |Q| \\ 0, \qquad u\ suprotnom \end{cases}$$

Primer:

Ukoliko imamo klasu C sa 3 metode, (M_1, M_2, M_3). I ukoliko je $\{I_1\} = \{a, b, c, d, e\}$, $\{I_2\} = \{a, b, e\}$ i $\{I_3\} = \{x, y, z\}$, tada je $\{I_1\} \cap \{I_2\}$ neprazni skup dok $\{I_1\} \cap \{I_3\}$ i $\{I_2\} \cap \{I_3\}$ su prazni skupovi. LCOM meri broj nepraznih skupova, što je u ovom slučaju 1.

Namena ove metrike je merenje kohezije klase, merenjem broja zajedničkih atributa koji se koriste u različitim metodama klase. Drugim rečima, ova metrika meri kvalitet apstrakcije koju klasa predstavlja. Visoka vrednost LCOM nagoveštava nedostatak kohezije, odnosno malu sličnost u okviru klase, tj. da se klasa sastoji od nepovezanih objekata. Velika kohezija u metodama klase je poželjna, pošto se ta klasa tada ne može podeliti i samim tim uslovljava da je enkapsulacija na visokom nivou. Mala kohezija može da signalizira greške prilikom procesa implementacije klase.

U [Henderson], izložena su dva problema vezana za ovu metriku:

- Dve klase mogu imati LCOM = 0, dok jedna ima više zajedničkih promenljivih, u odnosu na drugu.
- Ne postoje jasna uputstva kako da se LCOM vrednosti pravilno interpretiraju.

Stoga se u tom radu predlaže dopuna *LCOM* metrike. Ukoliko se razmatra skup metoda $\{M_I\}(I=1, ..., m)$, koje koriste skup atributa $\{A_j\}(j=1, ..., a)$ a broj metoda koje mogu da pristupe svakom atributu definišu kao $\mu(A_j)$, tada je:

$$LCOM^* = \frac{\left(\frac{1}{a}\sum_{j=1}^{a}\mu(A_j)\right) - m}{1 - m}$$

gde je LCOM* predložena unapređena LCOM metrika. Ukoliko sve metode pristupaju svim atributima, tada je $\sum_{j=1}^{a}\mu(A_j)=ma$, što znači da je LCOM* = 0, a to ujedno znači i da klasa ima savršenu koheziju. Ukoliko svaka metoda pristupa samo

jednom atributu, tada je $\sum_{j=1}^{a} \mu(A_j)=a$, što znači da je LCOM* = 1, a ta klasa pokazuje znake nedostatka kohezije. Vrednosti ovako dopunjene metrike, koje su blizu nule, imaće klase čije većine metoda pristupaju većini atributa, što ujedno i predstavlja znak visoke kohezije.

4.1.5 Metrike na nivou metoda klasa

Atributi i metode klasa se mogu smatrati najfinijim detaljima jedne klase. Ove karakteristike klasa su poznate, kao i tehnike koje ih analiziraju, ali metode klasa dodaju poseban nivo kompleksnosti koji se ogleda u činjenici da često metode jedne klase pozivaju metode druge klase. U OO sistemima, tradicionalne metrike kao što su broj linija programskog koda [Lorenz] (engl. **L**ines **O**f **C**ode, LOC) i kondicionalna kompleksnost [McCabe] (engl. **C**yclomatic **C**omplecity, CC) se najčešće koriste kao metrike na nivou metoda klasa za ocenu kvaliteta objektno orijentisane programske podrške. S obzirom na to da se metode klasa najčešće razvijaju na sličan način kao i funkcije u strukturalnom programiranju, one nisu toliko važne u OO dizajnu zbog sledećih razloga:

- najčešće su kratke i imaju manje uslovnih iskaza (*if* ili *case* iskazi).
- kompleksnost najčešće leži u komunikaciji između metoda a ne u okviru jedne metode.

Metrike koje mere kondicionalnu kompleksnost još uvek nisu značajno korišćene u OO sistemima. Ovo je iz razloga što će kompleksnost OO dizajna najčešće ležati u komunikaciji i interakciji metoda, a ne u strukturi jedne ili nekoliko metoda.

4.1.5.1 Broj linija programskog koda

Broj linija programskog koda [Lorenz] (engl. **L**ines **O**f **C**ode, LOC) predstavlja broj linija aktivnog programskog koda (programskog koda koji se izvršava) u okviru jedne metode. Ova metrika se koristi kao indikacija koliko je jednostavno razumeti, ponovo iskoristiti ili održavati programski kod, od strane programera. Granične vrednosti za procenu veličine koda zavise od programskog jezika i kompleksnosti same metode.

U OO sistemima, LOC bi trebalo da ima malu vrednost. U [Lorenz], predložen je prag u vrednosti LOC = 24, za C++ metode, a još manje za neke objektno orijentisane programske jezike sa dinamičkim tipovima, kakav je Smalltalk, LOC = 8. Ukoliko

metode sadrže veću vrednost LOC, tada predstavljaju dobre kandidate za deljenje u više manjih metoda.

Mana LOC metrike je da za istu funkcionalnost metode, varira u zavisnosti od programskog jezika koji se koristi, ali i kodnog stila, odnosno standarda. Ovu metriku nije preporučljivo koristiti kao samostalnu metriku prilikom ocene kvaliteta objektno orijentisane programske podrške. Ali zbog jednostavnosti i brzine računanja, i dalje je vrlo popularna prilikom analiza.

4.1.5.2 Broj poslatih poruka

Broj poslatih poruka [Lorenz] (engl. **N**umber **O**f **M**essages Send, NOM) meri broj poslatih poruka u okviru jedne metode, klasifikovanih po tipu poruka. Tipovi mogu biti:

- Unarni tip: poruke bez parametara
- Binarni tip: poruke bez parametara koje pripadaju specijalnom tipu
- Šifre: poruke sa jednim ili više parametara

NOM meri veličinu metoda na relativno nepristrastan način. U [Lorenz], predloženo je da se koristi granični prag u vrednosti od 9 za NOM. Treba voditi računa da programski jezici poput C++ mogu imati pozive ka delovima sistema koji nisu pisani korišćenjem objektno orijentisanih programskih jezika, pri čemu ovu komunikaciju ne treba računati u ukupan zbir poslatih poruka.

4.2 Mera kompletnosti programske sprege za servise digitalne televizije

Moderni digitalni TV prijemnici pružaju korisnicima širok spektar digitalnih servisa, od klasičnih TV servisa, baziranih na informacijama koje dolaze iz DVB transportnog toka, pa sve do modernih servisa kao što su Internet pretraživači, socijalne mreže ili VoIP usluge. Pružanje ovih modernih servisa korisnicima uređaja baziranih na Android OS, podržano je mogućnošću da se napredne aplikacije koje pružaju ovakve servise, preuzmu sa otvorenog marketa i instaliraju na krajnji uređaj. Ono što nedostaje Android OS je sistemska podrška za klasične servise digitalne televizije, kao što su prikaz i pretraživanje TV servisa, teletekst, prevod, elektronski programski vodič itd.

Prateći trendove na polju potrošačke elektronike, ali i modernih TV i STB uređaja, kao i literature koja opisuje karakteristike koje moderna TV-bazirana platforma mora da ispuni [Benoit], [Morris], [Noergaard], za potrebe izrade ove disertacije izdvojen je minimalan skup servisa digitalne televizije koje programska podrška treba da implementira. Tabela 4.1 pruža pregled ovi servisa, sa kratkim opisom, odnosno propratnim standardom koji ih detaljno opisuje. Odabrana lista servisa opisuje savremeni skup karakteristika prisutnih u modernim TV prijemnicima.

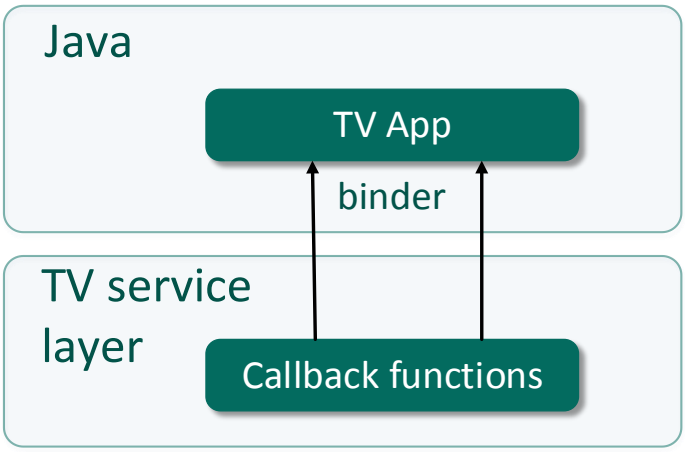
Ova lista servisa će biti korišćena i kao alat za proveru kompletnosti implementiranog programskog rešenja. Cilj ove disertacije je programska sprega koja, uz neophodna proširenja Android OS, implementira celokupan skup servisa digitalne televizije, opisan u tabeli ispod. Takođe, ova lista će biti korišćena i kao alat za poređenje komercijalno raspoloživih rešenja za integraciju TV servisa u Android OS, odnosno druge Java-bazirane platforme sa rešenjem predloženim u ovoj disertaciji. Cilj ovakve mere je da pokaže do kog stepena programska podrška koja se analizira ispunjava očekivanja sa stanovišta broja implementiranih TV servisa.

Funkcionalnost	**Skraćeni zapis**	**Detaljniji opis**
Prikaz i pronalaženje servisa	SCAN	Detaljan opis u standardu [ETSI2]
Elektronski programski vodič	EPG	Detaljan opis u standardima [ETSI3] i [ETSI5]
Prevod	SUB	Detaljan opis u standardu [ETSI4]
Teletekst	TTX	Detaljan opis u standardu [ETSI1]
Podsetnik	REM	Zakazivanje podsetnika za određene EPG događaje
Digitalni snimač	PVR	Detaljan opis u standardu [ETSI7]
Kontrolisani pristup	CA	Detaljan opis u standardu [ETSI6]
Roditeljska kontrola	PARENTAL	Detaljan opis u standardu [ETSI2]
Video na zahtev	VOD	Usluga video na zahtev koja omogućuje korisniku da odloženo gleda multimedijalni sadržaj. Najčešće vezana za IPTV uređaje.
Ulazne sprege	INPUT	Kontrola ulaznih audio/video sprega prisutnih na TV uređaju (kao što su HDMI, RGB, S-Video, CVBS, ...)

Tabela 4.1 Pregled izdvojenih servisa digitalne televizije

4.3 Mera kontrolabilnosti programske sprege za servise digitalne televizije

Kao jedna od važnih mera kvaliteta programske podrške za servise digitalne televizije, nameće se kontrolabilnost implementirane programske sprege. Na platformama baziranim na Androidu, razvoj aplikacija se zasniva na implementaciji GUI (engl. **G**raphical **U**ser Interface, **G**UI) dela aplikacije i pozadinskih servisa koji datu aplikaciju opslužuju informacijama za prikaz. Programska sprega koja implementira podršku za digitalnu televiziju takođe mora biti u formi pozadinskog procesa, odnosno Java servisa, kao što prikazuje Slika 4.2.



Slika 4.2 Arhitektura programske podrške u sistemu baziranom na Android OS

Kvalitet TV aplikacije koja se može razviti na Android OS, direktno zavisi od kvaliteta sprege Java servisa koji implementira DTV funkcionalnost. Jedan od važnih faktora prilikom ocene kvaliteta ovakvog Java servisa se ogleda u količini informacija, odnosno poruka koje se razmenjuju kroz slojeve programske podrške. U ovom slučaju, to je sloj gde se nalazi potencijalna TV aplikacija, i sloj gde se izvršava TV Java servis. Uzimajući u obzir količinu informacija koje obrađuje i broja stanja kroz koje prolazi sistemska TV programska podrška prilikom redovnog izvršavanja, da bi se implementirala TV aplikacija zadovoljavajućeg kvaliteta i karakteristika, potrebno je razmeniti srazmerno veliku količinu poruka.

Ove poruke se odnose na promene stanja, uspešne akcije, odnosno neuspešne operacije koje treba prikazati korisniku. Primeri za takve poruke su: promena stanja

prilikom pretrage frekventnog opsega za kanalima, uspešna promena kanala, neuspešno snimanje TV servisa, uspešan prikaz teletekst stranice, nemogućnost prikaza kanala usred roditeljske zaštite i mnoge druge.

Takođe, broj poruka će biti korišćen i kao alat za poređenje komercijalno raspoloživih rešenja za integraciju TV servisa u Android OS (ali i druge platforme bazirane na Java razvojnom okruženju), sa rešenjem predloženim u ovoj disertaciji. Cilj ovakve mere je da pokaže do kog stepena programska podrška koja se analizira ima mogućnost da ispuni očekivanja sa stanovišta broja poruka upućenih ka TV aplikaciji.

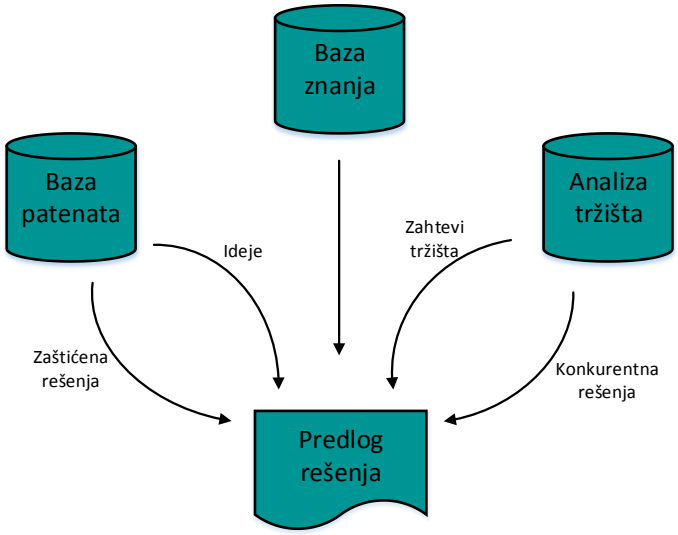
POGLAVLJE 5.

PROŠIRENJE ANDROID OPERATIVNOG SISTEMA SERVISIMA ZA DIGITALNU TELEVIZIJU

Motivacija za proširenje Android operativnog sistema servisima za digitalnu televiziju potiče od činjenice da moderni TV prijemnici prestaju korisnicima da pružaju isključivo tradicionalne TV servise i sve više pružaju korisnicima usluge koje nisu usko vezane za TV tehnologiju, kao što su Internet pretraživanje, VoIP komunikacija, video na zahtev ili socijalne mreže. Trenutni trend na tržištu modernih TV prijemnika je prelazak na Android operativni sistem, koji predstavlja bazu za sistemsku programsku podršku. Ovaj operativni sistem sadrži bogatu podršku za razvoj multimedijalnih, Internet baziranih ali i mnogih drugih tipova aplikacija koje trenutno vladaju tržištem aplikacija za mobilne uređaje.

Mana ovog operativnog sistema je što ne poseduje standardizovanu sistemsku podršku za servise digitalne televizije. Ovo rezultuje pojavom da na tržištu postoje industrijska rešenja, koja rešavaju ovaj problem, ali na nestandardizovan način ili način koji je prilagođen limitiranom skupu uređaja ili primena.

U cilju formiranja predloga rešenja, proučeni su raspoloživi izvori informacija.



Slika 5.1 Analiza relevantnih informacija (preuzeto iz [Papp])

Pretraga baze patenata rezultovala je skupom relevantnih patenata, ali pored toga dala i uvid u savremena istraživanja, i ukazala na neke nove ideje i trendove u datoj oblasti. U oblasti proširenja Android operativnog sistema servisima digitalne televizije, postoji porast aktivnosti poslednjih godina u pogledu zaštite intelektualne svojine. Svetski poznate kompanije su zainteresovane za ovu tehnologiju, što se vidi iz broja patenata iz te oblasti, što ujedno ukazuje i na aktuelnost i potencijal ovog pristupa.

U okviru analize tržišta tragalo se za rešenjima proširenja Android operativnog sistema servisima digitalne televizije, što je dovelo do formiranja skupa mogućnosti koje je potrebno podržati, ali su i identifikovani konkurenti i nedostaci postojećih rešenja. Pokazalo se da postoje komercijalna rešenja u odgovarajućem opsegu performansi, ali koja obezbeđuju samo određene funkcije (podržani su samo određeni servisi ili su rešenja usko specijalizovana za primenu, odnosno fizičku arhitekturu). Takođe, ne postoji standardizovano rešenje, koje bi bilo u mogućnosti da obezbedi potpunu funkcionalnost (koju opisuje Tabela 4.1) i da pri tome bude platformski nezavisno. Ovakvo rešenje bi omogućilo razvoj aplikacija optimizovanih za izvršavanje na TV uređajima i dalju spregu informacija koje dolaze iz DVB transportnog toka u Android ekosistem.

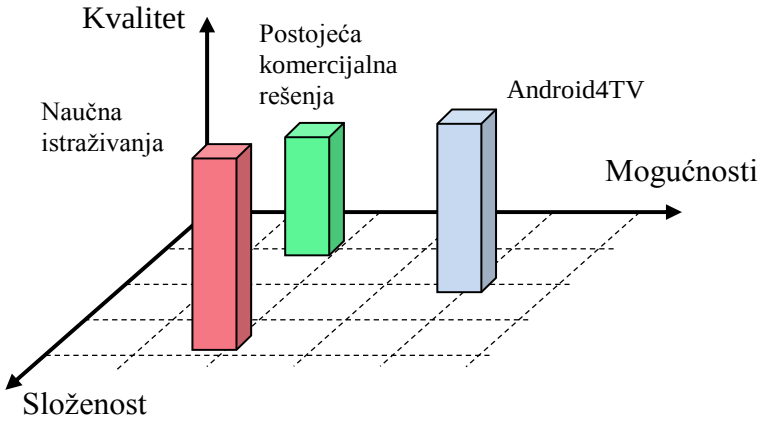
U disertaciji se, kao rešenje integracije digitalnih TV servisa u Android operativni sistem, predlaže skup programskih modula, koji su pozicionirani u svim programskim nivoima Android operativnog sistema, kao i skup pravila i preporuka za

izmenu jezgra operativnog sistema neophodnih za normalno funkcionisanje TV servisa.

Programski moduli predloženog programskog rešenja se nalaze podeljeni na dva dela:

- *Java programska sprega*: Platformski nezavisna programska podrška pisana u Java programskom jeziku, okrenuta ka TV aplikacijama, koja pruža unificiran pristup DTV resursima.
- *Izvršna programska sprega*: Platformski zavisna programska sprega okrenuta ka upravljanju elementima fizičke arhitekture i obradi u realnom vremenu za vremenski kritične operacije. Ova sprega je pisana u C/C++ programskom jeziku i dostupna je Java programskoj sprezi kroz JNI mehanizam.

Ovakav pristup garantuje platformsku nezavisnost predloženog rešenja kroz definiciju Java bazirane programske sprege, ali i performanse kroz obradu u realnom vremenu u modulima iz izvršnog nivoa. Očekuje se da se predloženo rešenje pozicionira među postojećim rešenjima kao što prikazuje Slika 5.2. Cilj je da se obezbedi bolji kvalitet u domenu kompletnosti i platformske nezavisnosti, ali da složenost sistema ne prelazi granice postavljene razvojem uređaja široke potrošnje. U toku istraživanja, usvojeno je ime koje imenuje razvijeno rešenje: *Android4TV*.



Slika 5.2 Formiranje zahteva i ograničenja

5.1 Neophodna proširenja jezgra Android operativnog sistema

Ovo poglavlje opisuje proširenja Android operativnog sistema koja su neophodna da bi se omogućilo upravljanje servisima digitalne televizije. Izmene opisane u ovom poglavlju predstavljaju proširenja jezgra operativnog sistema i kao takva ne mogu biti deo TV aplikacije. Ove izmene, sistemske po svojoj prirodi, zahtevaju da se na ciljnom uređaju, pre instaliranja predložene programske sprege, primeni prilagođena sistemska ROM slika (engl. **R**ead-**O**nly **M**emory system image).

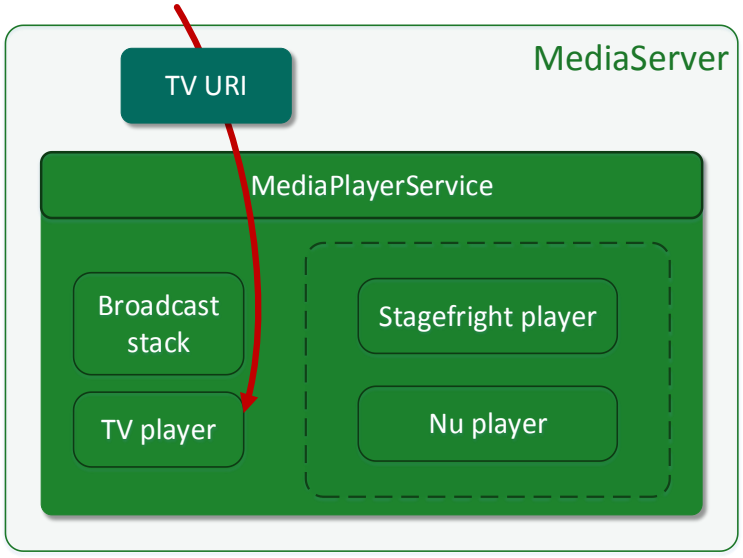
Prvi korak prilikom proširenja jezgra Android operativnog sistema je definisanje *View* elementa, koji će biti zadužen isključivo za prikaz video toka koji dolazi iz DVB transportnog toka. Android okruženje za prikaz video toka koristi posebnu grafičku komponentu *android.widget.VideoView*. Ova grafička komponenta je spregnuta sa podsistemom za prikaz multimedijalnih datoteka i različitih video tokova. S obzirom da je akcenat prilikom izrade disertacije bio na što manjim izmenama Android operativnog sistema, predloženo rešenje koristi isti multimedijalni tok kao i ostatak sistema. Za razliku od postojećeg podsistema, predloženo proširenje nema za cilj da obezbedi samo reprodukciju video datoteka iz lokalnog sistema datoteka, već i reprodukciju "živih" video izvora koji se često ne mogu apstrahovati datotekama (npr. video sprege kao što su HDMI, RGB, S-Video, CVBS).

Uzimajući u obzir da aplikativno okruženje Android operativnog sistema dozvoljava višestruko instanciranje *VideoView* (ili drugih nasleđenih) grafičkih komponenti u okviru jedne aplikacije, a u isto vreme, ugrađene platforme najčešće imaju ograničen broj video tokova koji mogu da se reprodukuju u paraleli, od izuzetne važnosti je uvođenje modula za upravljanje resursima. Ovo proširenje dozvoljava istovremeno i nesmetano deljenje resursa fizičke arhitekture između nekoliko aktivnih *VideoView* komponenti, čineći sistem svesnim karakteristika fizičke arhitekture na kojoj se izvršava. Namenske platforme koje poseduju veći broj aktivnih demultiplekser/deskrembler/dekoder/renderer komponenti fizičke arhitekture biće u mogućnosti da na najbolji način iskoriste svoj potencijal i postignu bolje performanse u odnosu na platforme sa ograničenim resursima.

5.1.1 Proširenja MediaServer modula

Jezgro Android multimedijalnog podsistema čini modul pod imenom *MediaServer*. Ovaj modul je, zapravo, izvršni (engl. native) pozadinski servis i instancira se među prvima prilikom pokretanja Android OS. Usled potrebe za pristupanjem Linux sistemskim uslužiocima (engl. device drivers) ovaj servis poseduje administratorska korisnička prava. Sam servis koristi okruženje *MediaPlayerService* za registraciju i kontrolu nad *MediaPlayer* objektima, koji su zaduženi za obradu multimedijalnih datoteka i njihov prikaz.

Da bi se uspostavo tok multimedijalnih podataka, *setDataSource* funkcija *VideoView* komponente se poziva sa odgovarajućim URI parametrom. Funkcija se dalje propagira kroz celu Android programsku arhitekturu, sve do *MediaServer* izvršnog posadinskog servisa koji definiše i obezbeđuje *MediaPlayer* objekat koji će biti zadužen za reprodukciju sadržaja prosleđenog od strane *VideoView* komponente.



Slika 5.3 Pozicija TVPlayer klijenta u MediaServer procesu

Za potrebe proširenja i prilagođenja Android OS servisima digitalne televizije, implementiran je poseban modul *MediaPlayer* tipa, pod nazivom *TVPlayer*. Ovaj modul je razvijen tako da se odaziva na posebno definisanu šemu TV URI identifikatora i uspostavlja vezu između Android TV aplikacije i izvršne TV implementacije.

5.1.2 TVPlayer modul

TVPlayer je instanca *MediaPlayer* klijenta u *MediaPlayerService* radnom okruženju. Namenjen je za inicijalizovanje, pokretanje, pauziranje i zaustavljanje TV MW obrade, ali i prezentovanje podataka koji su definisani od strane *setDataSource* URI parametra. Takođe se koristi i za preuzimanje *Surface* objekta namenjenog za video prezentaciju od strane TV MW. Postoje dva načina integracije *TVPlayer* instance u *MediaPlayerService* okruženje. Na Android platformama koje implementiraju OS verzije starije od 4.X, dodavanje nesistemskog klijenta je bilo jedino moguće izmenom programskog koda *MediaPlayerService* sistemskog servisa. Ova izmena je uključivala definisanje novog klijenta, kao i njegovu registraciju za određene URI tipove. Ovo je značajna izmena na sistemskom nivou Android OS i kao takva nije zanemarljiva prilikom sertifikacije uređaja koji je poseduju. Od verzije 4.1 omogućeno je dodavanje novog *MediaPlayer* klijenta bez izmena sistemskog programskog koda. Ovo je omogućeno *MediaPlayerFactory* mehanizmom, koji u toku izvršavanja sistema određuje koji prisutni klijenti će biti dodeljeni za opsluživanje određenih URI tipova, kao što je prikazano sledećim primerom.

```
class TVPlayerFactory : public MediaPlayerFactory::IFactory {
public:
    TVPlayerFactory(player_type playerType) {
        mPlayerType = playerType;
    }
    virtual float scoreFactory(const sp<IMediaPlayer> &client,
                               const char *url,
                               float curScore) {
        if (strncmp(url, "tv://", 5) == 0) {
            return 1.0f;
        } else {
            return 0.0f;
        }
    }
    virtual sp<MediaPlayerBase> createPlayer() {
        return new TVPlayer(mPlayerType);
    }
private:
    player_type mPlayerType;
};
```

Primer pokazuje upotrebu *MediaPlayerFactory* mehanizma prilikom registrovanja definisanog *TVPlayer* modula za odgovarajući URI tip, u ovom slučaju URI koji počinje sa "**tv://**". Iako izmena izvornog koda nije neophodna, neophodna je sistemska slika koja u sebi sadrži nesistemske klijente, pošto još uvek nije moguće instalirati klijente kao aplikacije sa marketa.

5.1.3 Jedinstveni TV identifikatori

TV URI format ima sledeću formu:

```
tv://<tv source>:<port number>?view=<view number>
```

Pri čemu **tv source**, može da ima jednu od sledećih vrednosti:

- dvb, atsc – za emitovani video
- hdmi, rgb, avin – za analogne/digitalne video ulaze

Brojanje portova (**port number** vrednost) kreće od 0. Vrednost za **view number** može da bude 0 ili 1, u zavisnosti od toga da li se koristi primarna ili sekundarna putanja za prikaz. Primarna putanja je namenjena da se koristi za prikaz video toka preko celog ekrana, dok je sekundarna namenjena za prikaz u PiP režimu. Ova promenljiva, zapravo, definiše audio/video putanju dekodera fizičke arhitekture. Ustaljeno je da integrisana kola (SoC) fizičke arhitekture (IC) poseduju različite mogućnosti dekodovanja i unapređenja slike u zavisnosti od toga da li se koristi primarni ili sekundarni A/V tok. Obezbeđivanjem ovog parametra, korisniku je dozvoljeno da utiče na odabir A/V toka za svaku *VideoView* komponentu iz same aplikacije.

Na primer, ukoliko se želi prikazati DVB video tok sa prvog DVB tjunera na primarnoj video putanji, potrebno je konstruisati sledeći TV URI:

```
tv://dvb:0?view=0
```

Drugi primer može biti prikaz video sadržaja drugog HDMI priključka na sekundarnoj video putanji (tipičan scenario korišćenja je prikaz sadržaja DVD izvora u PiP modu):

```
tv://hdmi:1?view=1
```

5.1.4 Upravljanje resursima (RM)

MediaPlayerService okruženje u Android OS programskoj arhitekturi ne poseduje podršku za zauzimanje, kontrolu i upravljanje resursima fizičke arhitekture (HW). Uzimajući u obzir činjenicu da fizičke arhitekture koje se najčešće koriste za razvoj TV/STB uređaja, poseduju veći broj raznolikih HW resursa, da bi se postiglo efikasno iskorišćenje potencijala ciljne platforme, neophodno je bilo proširi Android sistemski nivo posebnim modulom za upravljanje resursima (engl. **R**esource **M**anager, RM). Zahtevi postavljeni pred ovaj modul su:

- Mora da bude korišćen za upravljanje multimedijalnim resursima fizičke arhitekture (video/audio dekoderi i demultiplekseri).
- Da bude u potpunosti u skladu sa dinamičnom prirodom Android aplikacija (brza tranzicija između različitih aplikacija).
- Da održi punu kompatibilnost sa svim aplikacijama sa Android marketa.
- Da omogući integraciju TV sistemske programske podrške (TV MW) u Android programsku arhitekturu.

Jedan primer scenarija korišćenja TV uređaja baziranog na Android OS, gde značaj RM modula dolazi do izražaja, je kada korisnik ima unapred zakazano snimanje određene TV emisije, dok u isto vreme gleda multimedijalni sadržaj putem Interneta. Obe akcije zahtevaju isti tip resursa fizičke arhitekture, koji moraju biti usaglašeni između aplikacija koje ih traže. Na arhitekturama koje podržavaju gledanje i snimanje u isto vreme, ovo je izvodljivo, ali samo ukoliko postoji RM modul koji će voditi računa da obe akcije ne dođu u konflikt oko nekog resursa, već da pravilno rasporedi raspoložive resurse između njih.

5.1.4.1 Alokacija bazirana na prioritetima

RM modul u Android OS mora da koristi mehanizam za dodelu resursa koji je baziran na prioritetima. Ovo je neophodno da bi RM modul bio u stanju da odluči na pravilan način kom korisniku (aplikaciji) da oduzme resurse, a kom da ih dodeli. Da bi se ovo postiglo, poseban argument za prioritet se mora proslediti prilikom svakog zahteva za nekim resursom, kao što je prikazano u sledećem primeru.

```
resourceHolder = new ResourceHolder(clientName,
                                    callerPID,
                                    android::eRmPriorityRealTime);
resourceHolder->registerClient();
```

Na RM modulu je da odluči koja aplikacija u datom momentu najviše zaslužuje traženi resurs. Prioritet se podešava kada se klijent registruje ka RM. Najbolji primer potrebe za RM je glavna TV aplikacija. Ova aplikacija treba da je na vrhu liste prioriteta kada se dodeljuju resursi, i poslednja na listi aplikacija od koje se resursi oduzimaju. Ovim se postiže da TV uređaj na kojem se TV aplikacija izvršava uvek ima makar bazičnu TV funkcionalnost. Za druge aplikacije, prioritet će biti isti i niži od TV aplikacije. Ukoliko nekoliko aplikacija traže isti resurs, tada RM isti dodeljuje aplikaciji koja ga je poslednja zatražila. Ova pretpostavka se oslanja na činjenicu da poslednja aplikacija koja je zatražila resurs je ujedno i poslednja aplikacija koja je dobila komandu od strane korisnika uređaja, i samim tim, usluživanjem te aplikacije, korisnik stiče utisak bržeg odziva sistema. Sledeći primer pokazuje prethodno opisani mehanizam za alokaciju resursa.

```
sp<ResourceUser> Resources::findUserToPreemptFrom(
        ResourceType resourceType,
        sp<ResourceUser> resourceFutureUser) {
    ...
    for(size_t j = 0; j < instances.size(); j++) {
        const sp<Resource> &resource = instances[j];
        if(resource->mResourceState == kResourceState_InUse &&
                resource->mType == resourceType) {
            if(resource->mResourceUser->mPrio <= resourceFutureUser->mPrio) {
                resultUser = resource->mResourceUser;
                return resultUser; //user to preempt from
            }
        }
    }
    ...
}
```

5.1.4.2 Životni vek i način korišćenja RM

RM modul je predstavljen *singleton* implementacijom. Predložen tok poziva je:

1. Pokretanje
    a. Zajedno sa ostatkom Android sistemskih servisa
    b. Tokom inicijalizacije RM registruje sve resurse prisutne u sistemu

2. Registrovanje klijenata
    a. Klijenti dobavljaju instancu RM modula
    b. Pozivaju metodu za registraciju klijenta
3. Koraci za alokaciju resursa
    a. Aplikacija prvo proverava raspoloživost svih neophodnih resursa
    b. Ukoliko su svi resursi raspoloživi, RM modul dodeljuje resurse aplikaciji koja ih je tražila
    c. Ukoliko nema slobodnih resursa, ili nisu svi resursi slobodni, RM modul koristi sistem baziran na prioritetima da utvrdi od koje aplikacije je potrebno preuzeti resurs
        i. RM prosleđuje zahtev za preuzimanje resursa aplikaciji koja drži resurs pod kontrolom
        ii. Aplikacija ima određeno vreme da dati resurs oslobodi
        iii. Po oslobađanju resursa, RM dodeljuje isti aplikaciji koja ga je tražila

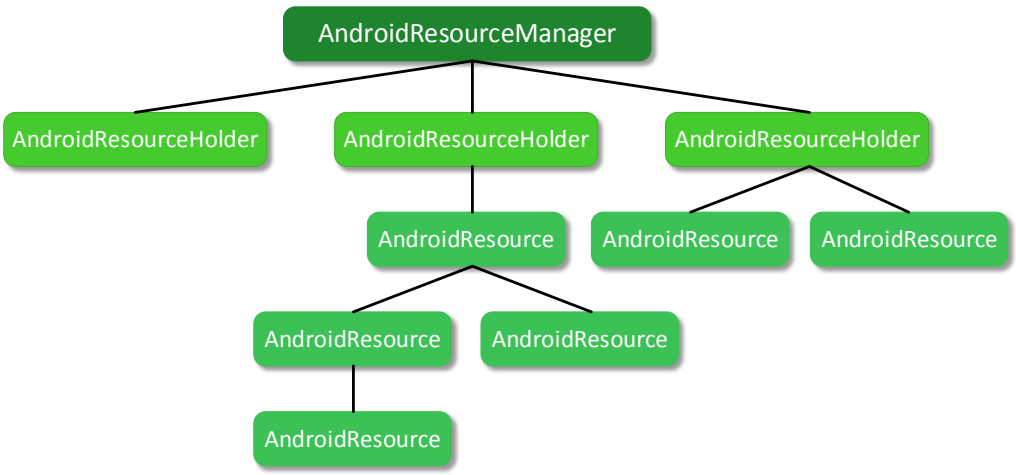
Sledeći primer pokazuje alokaciju neophodnih resursa od strane TV aplikacije, kao i odgovor na zahtev za oslobađanje zauzetog resursa.

```
mResourceHolder->requestResource(android::eVideoDec);
mResourceHolder->requestResource(android::eAudioDec);
mResourceHolder->requestResource(android::eDMX);
mResourceHolder->requestResource(android::eVideoRenderer);
mResourceHolder->requestResource(android::eAudioRenderer);
...
void TVPlayer::TVPlayerResourceManagerEventHandler::handleCallback(
        ResourceManagerEvent event) {
    switch(event) {
    case kStealing:
        ...
        pPlayer->releaseResources();
        break;
}
```

5.1.4.3 AndroidResourceManager objekat

AndroidResourceManager objekat se instancira kao *singleton* i služi za evidenciju raspoloživih resursa u sistemu i njihovih trenutnih korisnika. Svaki resurs ima vezu sa svojim vlasnikom, kao i vezu sa resursima od kojih direktno zavisi.

Slika 5.4 prikazuje instancu *AndroidResourceManager* sa tri registrovana klijenta. Prvi klijent ne zauzima nijedan resurs, dok drugi klijent koristi jedan resurs koji zavisi od još tri (na direktan i indirektan način). Treći klijent rezerviše dva resursa. Ovakva raspodela je moguća ukoliko nije došlo do konflikata prilikom raspodele traženih resursa i ukoliko je svaki klijent dobio sve resurse koje je tražio. Procedura razrešenja konflikata je opisana u sledećem pasusu.



Slika 5.4 Primer RM alokacije sa tri registrovana klijenta i resursima koje koriste

5.1.4.4 Zaustavljanje aplikacija

Za slučaj da se desi da iz nepoznatih razloga određena aplikacija ne oslobodi resurs u zahtevanom roku, RM modul može da zatraži da se data aplikacija nasilno zaustavi. Ovo je jedini način kako se osigurati da određeni resurs neće ostati "zaključan" ukoliko dođe do nepredviđenog ponašanja bilo koje aplikacije u sistemu. Da bi se odgovarajuća aplikacija mogla uspešno nasilno zatvoriti a da se zauzeti resursi ipak oslobode, RM modul vodi računa da uz svakog klijenta koji rezerviše resurs, sačuva i njegov PID. Sledeći primer pokazuje proces oslobađanja resursa iz ugla RM modula, koji uključuje slanje zahteva za oslobađanje resursa od strane aplikacije (opisano u prethodnom primeru), ali i nasilno zaustavljanje, ukoliko se resurs ne oslobodi u zahtevanom roku.

```
void ResourceManager::taskHandler(sp<Task> task) {
    ...
    case Task::kPreemptResource: {
        sp<ResourceUser> oldUser =
                resources->findUserToPreemptFrom(
                        getResourceType(),
                        resourceFutureUser);
        oldUser->resourceManagerEventCallback(kStealing);
        sleep(timeout); //add some time for releasing before kill
        ...
        kill(oldUser->mPID, SIGKILL);
        resources->setResourceState(resource, kResourceState_Available);
    }
}
```

5.2 Programska sprega za servise digitalne televizije bazirana na Java razvojnom okruženju

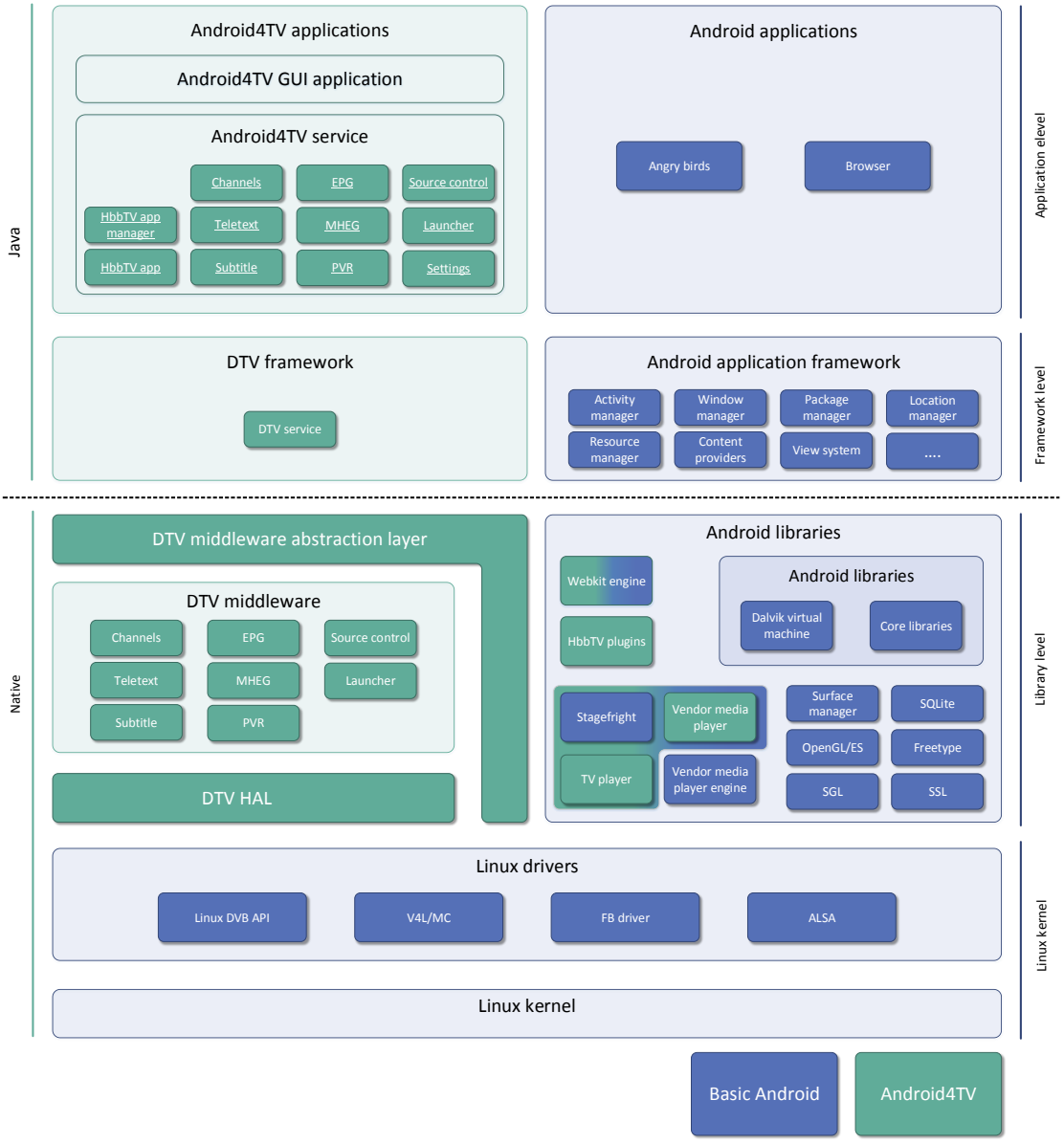
Programska sprega za servise digitalne televizije u predloženom rešenju je implementirana u Java programskom jeziku i pozicionirana u Android aplikativnom razvojnom okruženju. Ova sprega je okrenuta ka TV aplikacijama, kojima pruža unificiran pristup DTV resursima a pri tome garantuje platformsku nezavisnost i portabilnost. U toku istraživanja u okviru ove disertacije usvojeno je ime koje imenuje razvijeno programsko rešenje: *Android4TV*.

Ovo ime se ne odnosi isključivo na sloj programske sprege koja obezbeđuje TV servise, već na celokupan proces i predložen način integracije TV servisa u Android operativni sistem. Ovaj proces uključuje preporuke za izmenu i unapređenje jezgra Android operativnog sistema u domenu omogućavanja standardnih TV funkcija, potom proširenja standardnih Android mehanizama (kao što je predloženo proširenje IPC mehanizma podrškom za slanje klasne hijerarhije), i na kraju samu arhitekturu programske podrške za DTV.

5.2.1 Arhitektura predložene programske sprege (Android4TV)

Android4TV programsku spregu prikazuje Slika 5.5. Ova slika prikazuje ne samo predloženu programsku spregu, već i njenu poziciju i integraciju u odnosu na Android OS. Desnu stranu slike čine standardne komponente operativnog sistema, podeljene po tipu na Aplikacije (Java GUI i Java servisi), Aplikativno radno okruženje

(sistemski Java servisi), Sistemske biblioteke (C/C++) i Linux nivo (C/C++). Leva strane slike prikazuje predloženo rešenje, strukturno organizovano po istim nivoima. Slika takođe naglašava komponente jezgra Android operativnog sistema koje je bilo neophodno proširiti ili modifikovati u okviru razvoja rešenja.



Slika 5.5 Android4TV blok dijagram

Slika je, takođe, i vertikalno podeljena. Ova dimenzija podele naglašava programski jezik koji se koristi u datom programskom nivou Android arhitekture. Gornju polovinu čini sloj koji je implementiran u Java programskom nivou, i njega

čine sistemski Java servisi, sistemske Java aplikacije, ali i mesto gde se nalazi svaka naknadno instalirana aplikacija u sistemu. Donji sloj čini izvršni programski kod, pisan u C/C++ programskom jeziku, koji je u Android strukturi namenjen za sistemske biblioteke, ali i jezgro Linux operativnog sistema. Ovaj nivo je pretrpeo najveće izmene sa predloženim rešenjem, što je bilo i očekivano uzimajući u obzir da je ovaj nivo najviše u dodiru sa fizičkom arhitekturom i predstavlja jedino mesto gde se platformske izmene mogu primeniti. Takođe, ovaj nivo predstavljaju biblioteke koje zbog potrebe za pristupom rutinama niskog nivoa imaju posebna prava prilikom izvršavanja.

5.2.1.1 Android4TV aplikativni nivo

U Android4TV rešenju, aplikativni sloj se sastoji od dve Android aplikacije: *Android4TV GUI* i *Android4TVService*. Zajedno, ove dve aplikacije implementiraju funkcionalnost očekivanu od TV aplikacije koja se izvršava na modernom TV prijemniku (korišćenjem standardnih Android API i Android SDK skupova alata).

Android4TV GUI aplikacija implementira isključivo grafičku korisničku spregu TV aplikacije, korišćenjem Android grafičkih kontrola i mehanizama za razvoj GUI aplikacija. Ova aplikacija je, između ostalog, odgovorna za grafički prikaz živog video toka koji se emituje, ali i drugih podataka iz DVB transportnog toka, kao što su EPG, TTX, ili SUB.

Uloga *Android4TVService* aplikacije je isključivo da obezbedi pozadinski Java servis koji pruža podatke za prikaz grafičkoj sprezi implementiranoj u *Android4TV GUI* aplikaciji. Sama *Android4TVService* aplikacija ne implementira grafičku korisničku spregu (ne poseduje GUI elemente). Razlog uvođenja posebnog, pozadinskog Java servisa je dvojaka. Prvi razlog se ogleda u ograničenju nametnutom od strane Android okruženja za izvršavanje aplikacija (engl. Android runtime) koje ne dozvoljava aplikacijama koje implementiraju grafičku korisničku spregu da u svojim kontrolama vrše obradu (kao što je dobavljanje podataka) koje nije vremenski predvidivo. U ovakve obrade najčešće spada pristup spregama fizičke arhitekture, kao što su dobavljanje podataka putem Interneta (mrežna sprega) ili u domenu ove disertacije, pristup fizičkim DVB spregama. Razlog zašto je ovo ograničenje uvedeno u Android operativni sistem je da se spreči razvoj GUI aplikacija koje će imati spor

odziv na korisničke komande. Iz ovog razloga, svaka vremenski kritična obrada podataka se mora vršiti u pozadinskom Java servisu koji opslužuje GUI aplikaciju.

Drugi razlog postojanja *Android4TVService* pozadinskog Java servisa je vezan za činjenicu da moderni TV prijemnici, kroz svoje TV aplikacije, korisnicima nude napredne funkcije, koje predstavljaju nadogradnju funkcionalnosti standardnih digitalnih TV servisa i kao takve definišu ponašanje TV prijemnika. Jedan primer ovakve napredne funkcije može biti upravljanje odabirom audio toka (jezika) prilikom prikaza TV servisa. Standardna funkcionalnost u ovom slučaju predstavlja:

- Detektovanje prisutnih različitih audio tokova u jednom transportnom toku (TV servisu).
- Odabir audio toka za prikaz.
- Podešavanje podrazumevanog audio toka (koji će biti odabran ukoliko je prisutan u transportnom toku).

Definisanje ponašanja u slučaju kada podrazumevani audio tok nije pronađen u transportnom toku TV servisa koji se prikazuje, predstavlja naprednu funkciju (da li odabrati prvi raspoloživi jezik, ili ponuditi korisniku opciju da sam odabere).

Drugi primer je unapređenje funkcije istovremenog prikaza i snimanja TV servisa. Standardna funkcionalnost se ogleda u mogućnosti da se dva različita TV servisa, prisutna u listi servisa, u istom momentu obrađuju (jedan se prikazuje korisniku, dok se drugi snima na masovni medijum). U slučaju IPTV prijemnika, funkcija istovremenog prikaza i snimanja je ograničena propusnom moći Internet sprege koja je u tom momentu na raspolaganju. U slučaju kada je propusna moć dovoljno velika da obezbedi prikaz samo jednog IPTV servisa visoke definicije (HD), tada funkcija istovremenog prikaza i snimanja nije moguća usled nedovoljne propusne moći Internet sprege. Unapređenje ove funkcije se ogleda u tome da se u listi TV servisa pronađe IP servis koji je identičan servisu koji se trenutno prikazuje, ali u nižoj rezoluciji i nižoj bitskoj brzini (SD) i da se korisniku ponudi opcija da trenutni prikaz pređe na pronađen servis u nižoj bitskoj brzini, pri čemu bi se u tom slučaju, usled uštođenog protoka podataka omogućila funkcionalnost istovremenog prikaza i snimanja drugog video toka.

Ovakve i slične, napredne TV funkcije, opisane u prethodna dva primera, su usko specijalizovane za određene korisničke scenarije nametnute od strane

proizvođača TV prijemnika (kroz očekivanu funkcionalnost TV aplikacije). Iz tog razloga implementacija ovakvih naprednih funkcija, usled kompleksnosti obrade i činjenice da je usko vezana za TV aplikaciju, se nalazi u posebnom, pozadinskom Java servisu. *Android4TVService* predstavlja primer jednog takvog servisa koji koristi programsku spregu opisanu u narednom pasusu (koja ujedno i predstavlja glavni doprinos ove disertacije), za razvoj moderne TV aplikacije.

5.2.1.2 Android4TV okruženje za razvoj aplikacija optimizovanih za izvršavanje na TV uređajima (TVC)

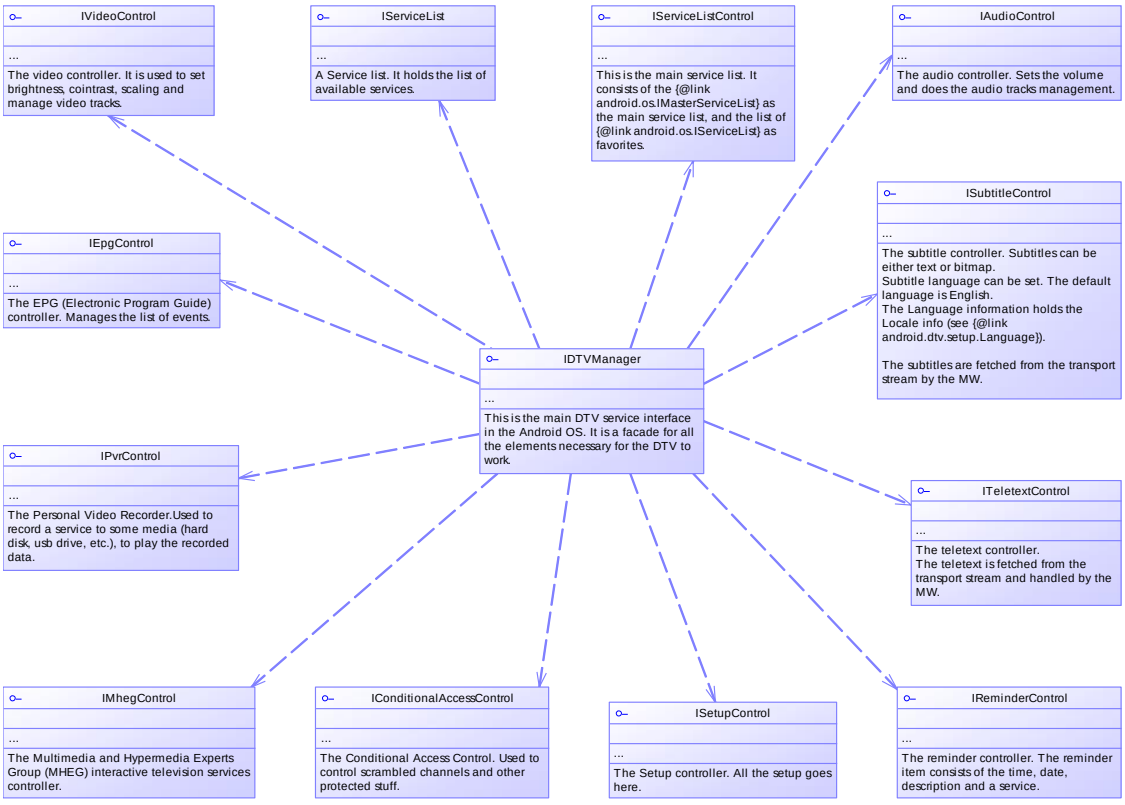
Na nivou aplikativnog radnog okruženja, Android4TV uvodi specijalizovani Java servis – *DTVService*. Ovaj servis je na raspolaganju ostatku Android sistema kroz standardni *Binder* mehanizam za međuprocesnu komunikaciju (IPC). *DTVService* enkapsulira sve DTV servise u Java klase, sprege (engl. interface) i enumeracije. Ovaj Java servis obezbeđuje standardizovan i unificiran Java API za pisanje proizvoljnih aplikacija optimizovanih za izvršavanje na TV uređajima (engl. **TV**-**C**entric applications, TVC).

DTV Java API se sastoji od sledećeg skupa Java paketa koji pokrivaju DTV funkcionalnost:

- *com.android.server*
- *android.dtv.audio*
- *android.dtv.ca*
- *android.dtv.callbacks*
- *android.dtv.epg*
- *android.dtv.io*
- *android.dtv.mheg*
- *android.dtv.ondemand*
- *android.dtv.parental*
- *android.dtv.picture*
- *android.dtv.pvr*
- *android.dtv.reminder*
- *android.dtv.route*
- *android.dtv.service*

- *android.dtv.setup*
- *android.dtv.sound*
- *android.dtv.subtitle*
- *android.dtv.swupdate*
- *android.dtv.teletext*
- *android.dtv.utils*
- *android.dtv.video*
- *android.os*

Paket *android.os* poseduje sve podržane sprege neophodne za razvoj DTV aplikacije. Slika 5.6 prikazuje sadržaj ovog paketa. Ostali paketi iz DTV Java API sloja sadrže klase koje implementiraju sprege definisane u *android.os* paketu. Za više detalja o samom DTV Java API nivou pogledati dodatak "Pregled DTV Java programske sprege".



Slika 5.6 Sadržaj android.os paketa

Kada se neka metoda iz *DTVService* Java servisa pozove od strane jedne od aplikacija, ona se prosleđuje JNI nivou, koji potom proziva odgovarajuću metodu iz izvršne biblioteke. Sa druge strane, za asinhrone događaje, koji se detektuju u programskom modulu izvršnog sloja, kada se događaj prosledi JNI nivou, on poziva odgovarajuću Java metodu iz *DTVService*. Ukoliko je potrebno, Java servis dalje proziva registrovanog klijenta (engl. listener) u Java aplikaciji.

Android4TV programska sprega je ujedno i razvojno okruženje koje omogućava programerima razvoj TV aplikacija koje su nezavisne od platforme na kojoj se izvršavaju. Time se postiže da jedna TV aplikacija može da se na jednak način i bez izmena izvršava na raznorodnim TV platformama, koje se najčešće značajno razlikuju kako po fizičkoj arhitekturi, broju i tipu ulaznih sprega, ali i raspoloživoj procesnoj moći.

5.2.1.3 Android4TV JNI sloj

Android4TV JNI sloj povezuje Java servise sa jedne strane i DTV implementaciju (DTV MW) iz izvršnog sloja sa druge. Ovaj sloj predstavlja most između Java i izvršnog C/C++ koda. On sadrži Java klase sa deklaracijom metoda koje su implementirane u C++ nivou. DTV JNI sloj se sastoji od sledećih klasa:

- *AudioControlNative*
- *VideoControlNative*
- *ConditionalAccessControlNative*
- *EpgControlNative*
- *MhegControlNative*
- *PvrControlNative*
- *ReminderControlNative*
- *ServiceListControlNative*
- *ServiceListNative*
- *SetupControlNative*
- *TimeDateNative*
- *AutoStandByTimerNative*
- *SleepTimerNative*
- *SubtitleControlNative*

- *TeletextControlNative*
- *InputOutputControlNative*
- *OnDemandControlNative*
- *ParentalControlNative*
- *PictureContro Native*
- *RouteControlNative*
- *ScanControlNative*
- *SoundControlNative*
- *SoftwareUpdateNative*

5.2.1.3.1 Mehanizam obradu asinhronih događaja

Osim što povezuje izvršni sloj i Java svet, ovaj sloj pruža i dvosmernu komunikaciju između DTV Java aplikacija i DTV MW. Prvi smer, od Java aplikacije do izvršnog sloja, implementiran je tako što Java aplikacija prvo proziva metodu iz Java servisa, koja potom proziva metodu iz JNI sloja, koja na kraju završava u DTV MW modulu. Svi pozivi u ovom smeru su sinhronizovani. U nastavku sledi primer ovog smera komunikacije, pokazan na slučaju iniciranja manuelne pretrage TV servisa iz DTV Java aplikacije. Komunikacija počinje u DTV Java aplikaciji:

```
IScanControl scanControl = dtvManager.getScanControl();
scanControl.manualScan(liveRoute, freq, updateList));
```

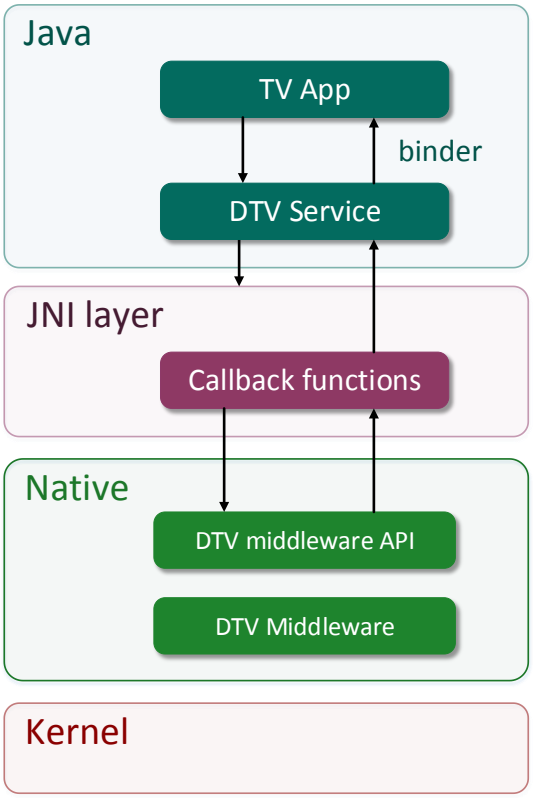
Ovi pozivi se potom prosleđuju *DTVService* Java servisu:

```
public class ScanControl extends IScanControl.Stub {
    public boolean manualScan(long installRouteID, long frequency,
            boolean updateList) throws RemoteException {
        return scanControlNative.manualScan(installRouteID,
            frequency, updateList);
    }
...
}
```

Java servis, potom proziva metodu iz JNI sloja, koja na kraju poziva odgovarajuće metode iz DTV MW modula:

```
JNIEXPORT jboolean JNICALL Java_android_dtv_service_ScanControlNative_manualScan
  (JNIEnv *env, jobject thiz, jlong installRouteID, jlong frequency, jboolean
updateList)
{
    DTV_MW_AppendList((mal_bool)updateList);
    DTV_MW_SetFrequency((mal_uint32)frequency);
    DTV_MW_ManualScan((mal_uint32)installRouteID);
    ...
}
```

Drugi smer komunikacije je kompleksniji. Kada se određeni DTV događaj desi, DTV MW proziva prethodno registrovanu metodu (engl. callback) iz JNI sloja. Potom, ovaj sloj proziva odgovarajuću metodu iz sloja Java servisa. Prozivanje Java metoda iz JNI sloja je asinhrono i da bi se uspešno izvršilo, JNI sloj mora da pronađe Java klasu koja implementira metodu koja treba da se prozove, nađe referencu na objekat te klase, identifikuje samu metodu i pozove je. Sve ovo se može postići korišćenjem *JNIEnv* objekta koji predstavlja vezu sa Java VM i može se dobaviti iz *JavaVM* objekta. Slika 5.7 prikazuje metod prozivanja metode Java servisa iz JNI sloja.



Slika 5.7 Dvosmerni mehanizam JNI sloja

U nastavku sledi primer ovog smera komunikacije, pokazan na slučaju prosleđivanja događaja iniciranog manuelnom pretragom TV servisa iz DTV MW izvršnog modula u DTV Java aplikaciju. Komunikacija počinje u DTV MW modulu:

```
getEnvRes=(*cached_jvm)->GetEnv(cached_jvm,(void**)&env, JNI_VERSION_1_4);
...
methodIDForCallback = (*env)->GetStaticMethodID(env,
    globalClazCallbacks, "callbackCall", "(II)V");
...
(*env)->CallStaticVoidMethod(env, globalClazCallbacks,
    methodIDForCallback, (jint)typeOfCallback, (jint)callbackValue);
```

DTV MW modul poziva metodu *callbackCall* iz *DTVService* Java servisa. Ova metoda, potom poziva prethodno registrovanu metodu iz klase koja se nalazi u DTV Java aplikaciji:

```
public static void callbackCall(int typeOfCallback, int callbackValue) {
    switch (typeOfCallback) {
    case SCAN_COMPLETE:
        channelsCallback.scanFinished();
        break;
    ...
    }
}
```

Na kraju, da bi komunikacija bila uspešna, DTV Java aplikacija mora da implementira sledeće korake:

```
private IChannelsCallback channelCallback = new IChannelsCallback.Stub() {
    public void scanFinished() throws RemoteException {
        ...
    }
}
...
ISetupControl setupControl = dtvManager.getSetupControl();
setupControl.setChannelsCallback(liveRoute, channelCallback.asBinder());
```

Najvažniji korak je registrovanje metode koja će biti prozvana kada se odgovarajući događaj registruje (*setChannelsCallback* metoda).

5.2.1.4 Android4TV izvršni sloj

Android4TV izvršni sloj predstavlja mesto koje je namenjeno za DTV programsku podršku niskog nivoa. Ova programska podrška je izrazito platformski

zavisna, pisana u C/C++ jeziku i služi za direktnu manipulaciju sa resursima fizičke arhitekture i to:

- Audio/video dekoderi
- Tjuneri/demodulatori
- Ulazne audio/video sprege
- Demultiplekseri

Ovaj sloj se svim višim slojevima prikazuje putem programske sprege koja ima za cilj da sakrije platformske specifičnosti. Osim što od viših slojeva krije specifičnosti fizičke arhitekture ciljne platforme, ovaj sloj od viših, platformski nezavisnih slojeva krije i tip i arhitekturu izvršne DTV programske sprege (DTV MW). Ovim pristupom je omogućeno da se ista TV aplikacija bez izmena izvršava na različitim platformama.

5.2.2 Prilagođenje IPC komunikacije sa stanovišta nasleđivanja kompleksnih objekata

Problem se javlja prilikom prosleđivanja atributa i metoda klasa naslednica kroz Android *Binder* IPC mehanizam u Android4TV okruženju. Naime, kroz Android *Binder* nije moguće proslediti atribute i metode klasa naslednica pa se morao pronaći adekvatan mehanizam za prosleđivanje istih, s obzirom na to da je u Android4TV programskoj podršci neophodno prikazati sve informacije iz klasa naslednica, a ne samo roditeljske klase.

Android *Binder* može da prenese primitivne tipove i string objekte bez ikakve intervencije. Međutim, ako je potrebno kroz Android *Binder* propustiti proizvoljni objekat bilo koje klase onda se mora implementirati dodatna sprega, *Parcelable*. Da bi se implementirala ova sprega neophodno je dodati dve nove metode (metodu *describeContents()* i metodu *writeToParcel()*) kao i kompleksni atribut *CREATOR*. Metode *describeContents()* i *writeToParcel()* postoje u sprezi *Parcelable* pa se zbog toga moraju implementirati da bi sprega uopšte funkcionisala. Metoda *describeContents()* vraća celobrojnu vrednost čiji bitovi opisuju specijalni sadržaj klase. U slučaju kada se koristi polimorfizam, ova metoda može da se koristi za prenos informacije kog je tipa klasa naslednica, i da se po prijemu objekta ove klase pozove odgovarajući konstruktor. Ovaj pristup se pokazao kao nedovoljno fleksibilan jer zahteva definisanje i ručno dodavanje posebnog slučaja za svaku novu klasu

naslednicu u okviru programskog koda roditeljske klase. Predloženo rešenje omogućuje automatizaciju ovog postupka, korišćenjem mehanizma refleksije, pri čemu nisu potrebne izmene ni u roditeljskoj, ni u klasi naslednici. Pored datih metoda sprege, predloženo rešenje uključuje i metodu *readFromParcel()*. Ova metoda nije uobičajena, ali se koristi da se na jednom mestu pročitaju vrednosti iz Android *Binder* prenosa i da se na osnovu toga inicijalizuju svi atributi klase. Metodu *readFromParcel()* poziva klasa naslednica. Predloženo rešenje će biti pokazano kroz primer implementacije jedne roditeljske klase (klasa *Content*) i jedne klase naslednice (klasa *ServiceContent*).

```
public class Content implements Parcelable {
    public int describeContents() {
        return 0;
    }
    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(className);
        dest.writeInt(index);
        dest.writeString(name);
        dest.writeString(image);
        ...
    }
    public static final Parcelable.Creator<Content> CREATOR
            = new Parcelable.Creator<Content>() {
        public Content createFromParcel(Parcel in) {
            String className = in.readString();

            try {
                Class clazz = Class.forName(className);
                Constructor c = clazz.getDeclaredConstructor(Parcel.class);
                Content retVal = (Content) c.newInstance(in);
                Field classNameField = clazz.getField("className");
                classNameField.set(retVal, className);
                return retVal;
            } catch (... e) {
            }
            return null;
        }
    };
    ...
}
```

U prikazanom primeru, klasa *Content* predstavlja roditeljsku klasu i sadrži izvestan broj atributa (npr. index, name, image…), predloženo proširenje u vidu

atributa *className* za prenos klasne hijerarhije kroz Android *Binder* kao i jedan dodatni atribut *CREATOR*. Ovaj atribut implementira spregu tipa *Parcelable* i sadrži metodu *createFromParcel()*. Metoda *createFromParcel()* se poziva kada željeni objekat pristigne na odredište (programska podrška koja ga koristi) kroz Android *Binder* mehanizam. Ova metoda poziva konstruktor koji prima *Parcel* objekat. Da bi se prosledile prave informacije do aplikacije, neophodno je odrediti koji konstruktor se poziva, odnosno konstruktor koje klase naslednice. Učitavanje je postignuto tako što je prvo učitan pun naziv klase uz pomoć standardne metode *readString()*. Koristeći refleksiju kreira se objekat klase naslednice i poziva njegov konstruktor, koji radi sa Android *Binder* mehanizmom, odnosno preuzima podatke iz Android *Binder* poziva. U nastavku je pokazan ovaj mehanizam na primeru klase *ServiceContent*.

```
public class ServiceContent extends Content implements Parcelable {
    ...
    public static final Parcelable.Creator<ServiceContent> CREATOR
            = new Parcelable.Creator<ServiceContent>() {
        public ServiceContent createFromParcel(Parcel in) {
            return new ServiceContent(in);
        }
    };
    public ServiceContent(Parcel in) {
        readFromParcel(in);
    }
    public void writeToParcel(Parcel dest, int flags) {
        super.writeToParcel(dest, flags);
        dest.writeByte((byte) (sendService ? 1 : 0));
    }
    public void readFromParcel(Parcel in) {
        super.readFromParcel(in);
        this.sendService = in.readByte() == 1;
    }
    ...
}
```

S obzirom da Android *Binder* predstavlja mehanizam za prenos objekata između dva procesa, prvi proces će kreirati objekat klase naslednice *ServiceContent* i poslaće ga drugom procesu. Ako posmatramo poziv metode drugog procesa, onda je moguće deklarisati metodu u drugom procesu na dva načina. Prvi način je kreiranje odgovarajuće metode za svaku klasu naslednicu (*getServiceContent*, *getRadioContent*), dok drugi način omogućuje da se koristi polimorfizam, čime se dobija samo jedna metoda koja prima objekat roditeljske klase, što je i cilj predloženog rešenja. Dakle,

kada objekat klase naslednice stigne na odredište, Android *Binder* će ga tretirati kao da je objekat roditeljske klase *Content*.

Poznato je da Android *Binder* nije u stanju da uvaži celo stablo nasleđivanja i da kreira odgovarajući objekat klase naslednice, ali se ovaj problem rešava već pomenutim pozivom metode *createFromParcel()*, pri čemu se zna da je Android *Binder* pokrenuo prenos roditeljske klase *Content*. Na ovaj, način Android *Binder* prvo preuzima ime klase i napravi objekat odgovarajuće klase naslednice. Zatim se poziva konstruktor koji prima *Parcel*, konstruktor *public ServiceContent(Parcel in)*. Ovaj konstruktor potom poziva metodu *readFromParcel()*, koja poziva roditeljsku metodu istog imena iz klase *Content*, a zatim čita još jedan dodatni bajt iz Android *Binder* prenosa, koji pripada isključivo klasi naslednici, *ServiceContent*. Svaka klasa naslednica, svakako poziva metodu *readFromParcel()* koja učitava sve atribute iz klase *Content*, dok se učitavanje dodatnih atributa vrši dodavanjem metode *readFromParcel()* u klasi naslednici, pri čemu je neophodno pozvati roditeljsku metodu *super.readFromParcel()* kako bi se svi osnovni atributi učitali. Sa druge strane, slanje objekata vrši se na sličan način. Naime, kada je napravljena klasa *ServiceContent*, objekat se prosleđuje Android *Binder* mehanizmu. Stvoreni objekat klase *ServiceContent* već ima podešen atribut *className* na njegov pun naziv. Android *Binder* poziva metodu *writeToParcel()*, a u njoj i roditeljsku metodu *super.writeToParcel()*, a zatim šalje dodatne atribute koji pripadaju samo klasi naslednici, *ServiceContent*.

Postojeći mehanizam prenosa klasa naslednica se bazira, kao što je to već rečeno, na realizaciji metode *describeContents()*. Ovo rešenje nije fleksibilno, pošto se za svaku novu klasu naslednicu mora modifikovati roditeljska klasa, kao što je prikazano u sledećem primeru.

U primeru se može videti da metoda *describeContents()* vraća celobrojnu vrednost čiji bitovi opisuju specijalni sadržaj klase, pri čemu su za isti primer prenosa klasa naslednica *Content* klase definisane sledeće vrednosti:

- **1**: za *Content*:
- **2**: za *ServiceContent*
- **3**: za *RadioContent*

```
public static final Parcelable.Creator<Content> CREATOR
        = new Parcelable.Creator<Content>() {
    public Content createFromParcel(Parcel in) {

    public Content createFromParcel(Parcel in)
    {
        int description=in.readInt();
        switch(description)
        {
            case 1:
                return new Content(in);
            case 2:
                return new ServiceContent(in);
            case 3:
                return new RadioContent(in);
            default:
                return null;
        }
    }
}
```

Iz primera se može zaključiti da prilikom korišćenja standardnih Android mehanizama za prenos, u roditeljskoj klasi se mora definisati poseban slučaj za svaku novu klasu koja se doda u klasnu hijerarhiju roditeljske klase. Predloženo rešenje u ovoj disertaciji omogućuje automatizaciju ovog procesa, pri čemu se u roditeljskoj i klasi naslednici ništa ne menja, s obzirom da se svi atributi i objekti u klasi naslednici nasleđuju od roditeljske klase.

5.2.3 Pregled Android4TV programskih modula

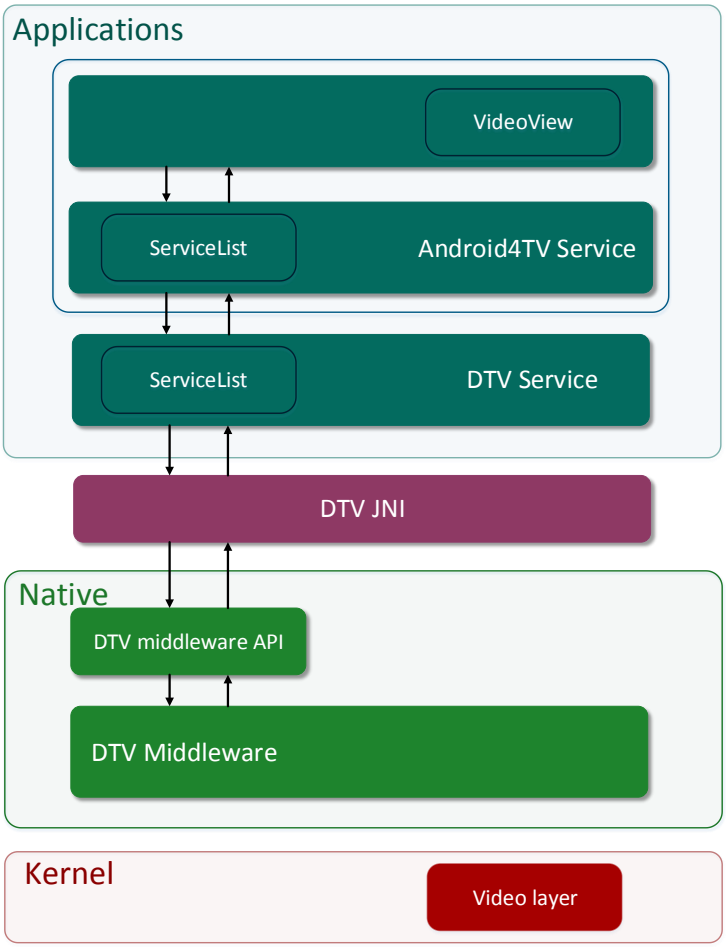
Android4TV programsko rešenje se sastoji od modula programske podrške koji pokrivaju osnovne DTV funkcije prisutne na modernim TV/STB prijemnicima. Ovi moduli su:

- Prikaz emitovanog signala
- Teletekst modul
- Modul za prikaz prevoda
- Elektronski programski vodič
- Lični video snimač
- Uslovni pristup/prikaz

Implementacija svakog od ovih modula uključuje programsku podršku u svim prethodno spomenutim nivoima programske podrške koji čine arhitekturu Android4TV rešenja. Detaljniji opis funkcionalnosti i strukture svakog od modula sledi u narednim poglavljima.

5.2.3.1 Prikaz emitovanog signala

Prilikom implementacije modula za prikaz DVB emitovanog signala, potrebno je implementaciju propagirati kroz sve nivoe Android4TV programske podrške. U ovom slučaju to uključuje TV aplikaciju, njen Java servis, potom *DTVService*, JNI sloj, ali i izvršni (engl. native) sloj programske podrške, kao što i prikazuje Slika 5.8.



Slika 5.8 Arhitektura modula za prikaz DVB emitovanog signala

Da bi prikazala sliku, TV aplikacija instancira *VideoView* grafičku komponentu, čija je glavna uloga da uspostavi multimedijalni video tok, i omogući prikaz sadržaja

video ravni. Pomoću TV URI parametra, aplikacija može da utiče na izvor video signala, što će u ovom slučaju biti "dvb", ali i video putanja (primarna, sekundarna). Osim obezbeđivanja video toka, potrebno je i DTV MW modulu naložiti da prikaže odgovarajući TV servis iz liste raspoloživih servisa. Ovo se postiže pozivanjem metoda iz *DTVService* modula putem *Binder* IPC mehanizma.

DTVService modul tada može putem JNI sloja da pokrene prikaz zatraženog TV servisa, korišćenjem metoda DTV MW modula (koji se nalazi u Android4TV izvršnom sloju). DTV MW modul dalje korišćenjem resursa fizičke arhitekture i to, tjunera, demultipleksera, dekodera i renderera, obezbeđuje pozicioniranje na odgovarajuću frekvenciju, parsiranje transportnog toka u elementarne audio/video tokove, njihovo dekodovanje i, na kraju, prikaz TV servisa u grafičkoj jedinici. Za napredne TV aplikacije, koje žele da implementiraju PiP funkcionalnost, postupak je identičan, s tim da manji video prozor se pozicionira i smanjuje korišćenjem metoda *VideoView* komponente. Ovo je izvodljivo jedino pod pretpostavkom da ciljna fizička arhitektura ima mogućnost da prikaže više video tokova istovremeno.

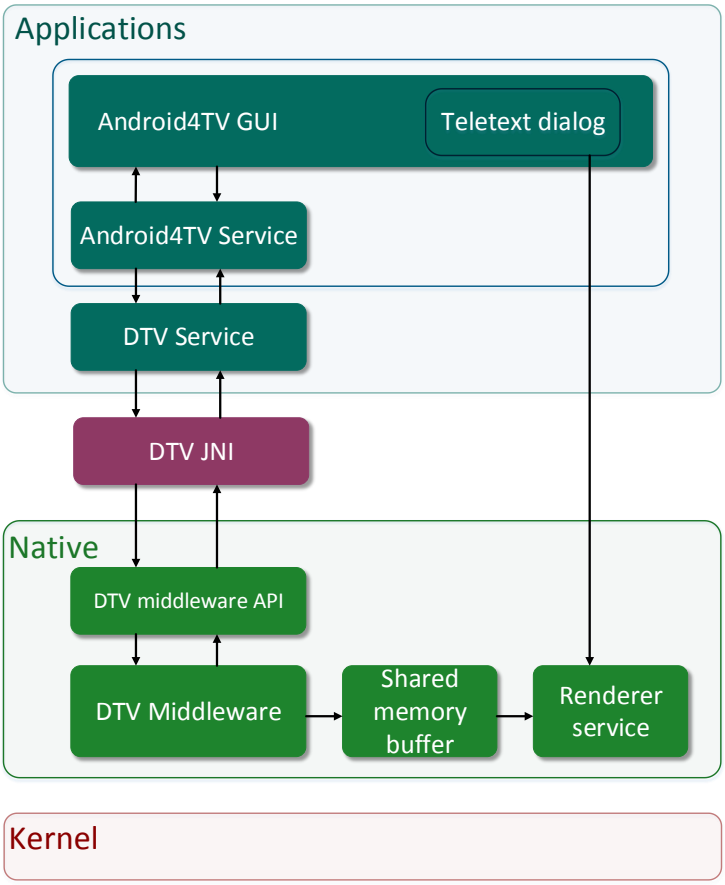
Modul za prikaz DVB emitovanog signala implementira sledeće funkcionalnosti:

- Prikaz "živog" video toka – audio/video signal izvučen iz trenutnog izabranog kanala.
- Pronalaženje TV servisa – pretraga frekventnog opsega i prikupljanje informacija o raspoloživim TV servisima.
- Upravljanje listom raspoloživih TV kanala – dodavanje/brisanje, pravljenje lista "omiljenih" (engl. favourites) kanala.
- Preključivanje na izabrani kanal – odabir i prikaz odabranog kanala.
- Podešavanje slike i zvuka – napredna podešavanja kvaliteta slike i zvuka.
- Aplikativna podešavanja – promena zemlje, vremenske zone, jezika ili druga regionalna podešavanja

5.2.3.2 Teletekst modul (TTX)

Prilikom implementacije modula za prikaz teleteksta, potrebno je implementaciju propagirati kroz sve nivoe Android4TV programske podrške. U ovom slučaju to uključuje TV aplikaciju, njen Java servis, potom *DTVService*, JNI sloj, ali i izvršni sloj programske podrške, kao što prikazuje Slika 5.9.

Specifičnost prikaza teletekst stranica se ogleda u tome što komponenta koja vrši dekodovanje i konstruisanje stranice na osnovu informacija iz transportnog toka (za više informacija o tome kako se teletekst podaci prenose u DVB transportnom toku, pogledati dodatak "Servisne informacije (SI) DVB standarda") nema informaciju o tome gde da konstruisanu stranicu prikaže. Ovo je zbog toga što se kontrola grafičkog prikaza može izvršiti samo iz TV aplikacije. Za tu namenu, implementiran je poseban izvršni pozadinski servis (*Renderer service*, RS, sa slike), koji predstavlja sponu između TV aplikacije, odgovorne za obezbeđivanje i kontrolu grafičkih komponenti i DVB MW, odgovornog za dobavljanje i prikaz željenih podataka. TV aplikacija prosleđuje ka RS dijalog u koji želi da se iscrta teletekst strana, dok sam DVB MW, putem RS dolazi do pokazivača na memorijsku zonu gde treba da generiše teletekst stranu. Za komunikaciju i deljenje memorijskih zona između ova dva procesa, koristi se standardni Android mehanizam – ASHMEM [Yaghmour].



Slika 5.9 Mehanizam prikaza teletekst stranice

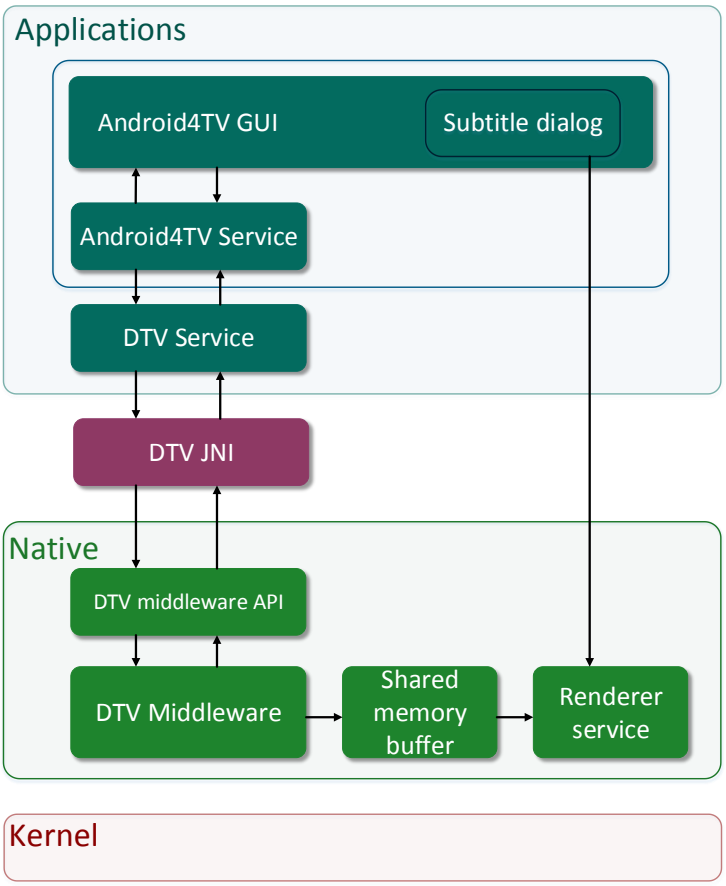
5.2.3.3 Modul za prikaz prevoda (SUB)

Modul za prikaz prevoda je po svojoj arhitekturi identičan modulu za prikaz teleteksta. DVB MW je zadužen za izdvajanje informacija o prevodu iz DVB transportnog toka, generisanje i skladištenje u deljenu memorijsku zonu. TV aplikacija zauzima grafički objekat za prikaz prevoda na ekranu, dok *Renderer service* izvršni modul služi kao spona između prethodna dva modula. Jedina razlika između TTX i SUB modula je veličina deljenog bafera, ali i, što je još važnije, broj, tip i dinamika poruka razmenjenih kroz Android4TV programsku arhitekturu. Ove razlike su uslovljene razlikama u samim TTX i SUB digitalnim servisima i biće detaljnije objašnjene u nastavku. Dinamika prikaza prevoda nije diktirana korisničkim komandama, kao u slučaju teleteksta, već se određuje na osnovu informacija prisutnih u DVB transportnom toku i stoga je potrebno definisati posebne poruke koje se razmenjuju u slučaju potrebe za prikazom prevoda. Takođe, prevod je potrebno iscrtati liniju po liniju (za razliku od teleteksta koji se iscrtava stranicu po stranicu) i potrebno ga je ukloniti posle određenog vremenskog perioda validnosti (za razliku od teleteksta koji se uklanja samo na korisnički zahtev). Na kraju, količina podataka potrebna za skladištenje jedne teletekst stranice je fiksne veličine, dok količina podataka potrebnih za skladištenje jedne linije prevoda zavisi od toga da li je u pitanju prevod za TV servis visoke (HD) ili standardne (SD) rezolucije. Slika 5.10 prikazuje mehanizam prikaza prevoda.

5.2.3.4 Elektronski programski vodič (EPG)

Prilikom implementacije modula za elektronski programski vodič, potrebno je implementaciju propagirati kroz sve nivoe Android4TV programske podrške. U ovom slučaju to uključuje TV aplikaciju, njen Java servis, potom *DTVService*, JNI sloj, ali i izvršni sloj programske podrške.

TV aplikacija obezbeđuje grafičke komponente za prikaz elektronskog programskog vodiča. Ova aplikacija prikazuje EPG podatke dobijene od nižih slojeva programske arhitekture ali i prikuplja ulaze od korisnika (za prikaz narednog EPG podatka, ili EPG podataka za drugi TV servis ili drugi vremenski period).



Slika 5.10 Mehanizam prikaza stranice prevoda

Paket *android.dtv.epg*, koji je deo *DTVService* sloja, enkapsulira EPG funkcionalnost koja stiže iz izvršnog sloja (JNI i DTV MW) u Java objekte koje obezbeđuje TV aplikaciji.

Sama obrada EPG podataka, njihovo izdvajanje i parsiranje iz DVB transportnog toka se izvršava u vremenski kritičnom DTV MW izvršnom sloju. EPG informacije se dobavljaju iz EIT tabela MPEG-2 transportnog toka (više detalja o ovom procesu se nalaze u dodatku "Servisne informacije (SI) DVB standarda"). Predloženo programsko rešenje je u stanju da obezbedi TV aplikaciji EPG događaje u periodu od 7 dana unapred od momenta prikaza. Osim prostog prikaza, TV aplikacija ima i mogućnost da kroz *DTVService* API i zakaže odloženo snimanje (više o ovoj funkcionalnosti u narednom poglavlju).

5.2.3.5 Lični snimač video signala (PVR)

Modul za snimanje video sadržaja (lični video snimač), kao i svi drugi moduli Android4TV rešenja, svoju implementaciju ima u nekoliko programskih slojeva. Prvi deo, izložen korisniku, nalazi se u okviru TV aplikacije i ogleda se u grafičkim elementima koji omogućavaju korisniku upotrebu PVR funkcionalnosti. Na *DTVService* nivou, PVR funkcionalnost je sadržana u *android.dtv.pvr* Java paketu koji, putem JNI podrške, komunicira sa vremenski kritičnom implementacijom u okviru DTV MW sloja.

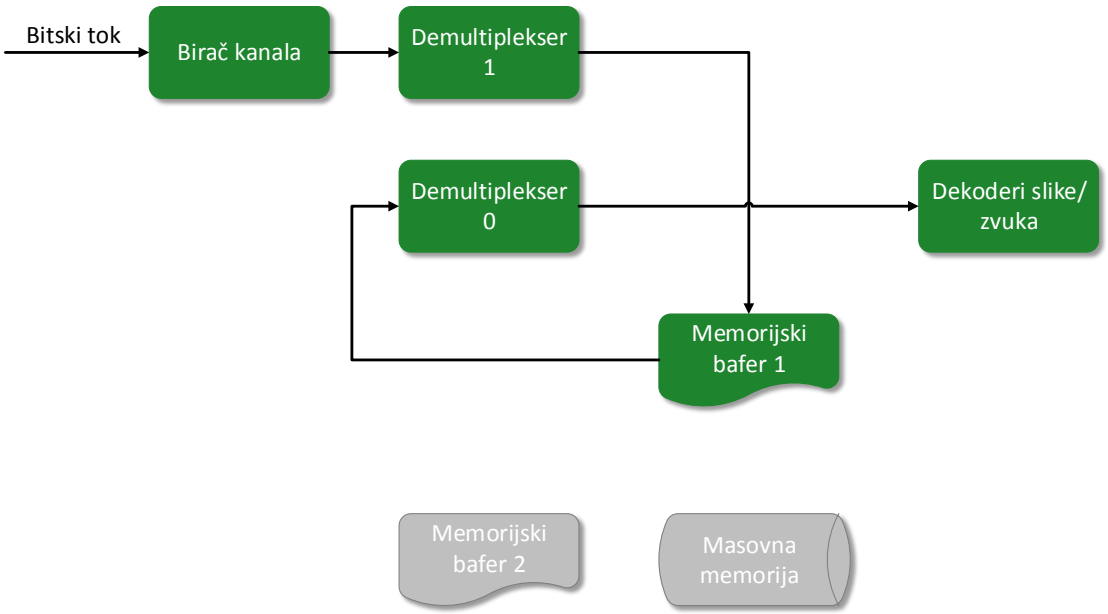
Funkcionalnost koja se očekuje od svakog modernog TV prijemnika koji podržava PVR funkcionalnost je:

- Prikaz liste snimljenih emisija
- Pokretanje snimanja servisa
- Zaustavljanje snimanja servisa
- Istovremeno snimanje i gledanje TV servisa
- Odloženo gledanje servisa
- Zakazivanje snimanja
- Upravljanje uređajima za skladištenje podataka

Da bi se postigla celokupna funkcionalnost PVR modula, potrebno je izvršiti izmene modula koji vrši usmeravanje DVB transportnog toka, jer u ovom slučaju, osim jedne jednostavne putanje transportnog toka, koja je potrebna prilikom prikaza jednog TV servisa, u slučaju PVR funkcionalnosti, potrebno je dodati i modul memorijskog bafera koji će služiti za odloženi prikaz i snimanje. Više o neophodnim izmenama u nastavku.

5.2.3.5.1 Prikaz uživo emitovanog servisa

Način prikaza jednog servisa uživo se u PVR režimu obavlja na identičan način kao i van PVR režima, ukoliko se posmatraju samo viši slojevi programske podrške televizijskog prijemnika. Veoma bitna razlika odnosi se na način usmeravanja podataka, tako da su za uživo emitovanje servisa u PVR režimu potrebni sledeći elementi, i to na način koji prikazuje Slika 5.11.



Slika 5.11 Putanja podataka kod emitovanja servisa uživo

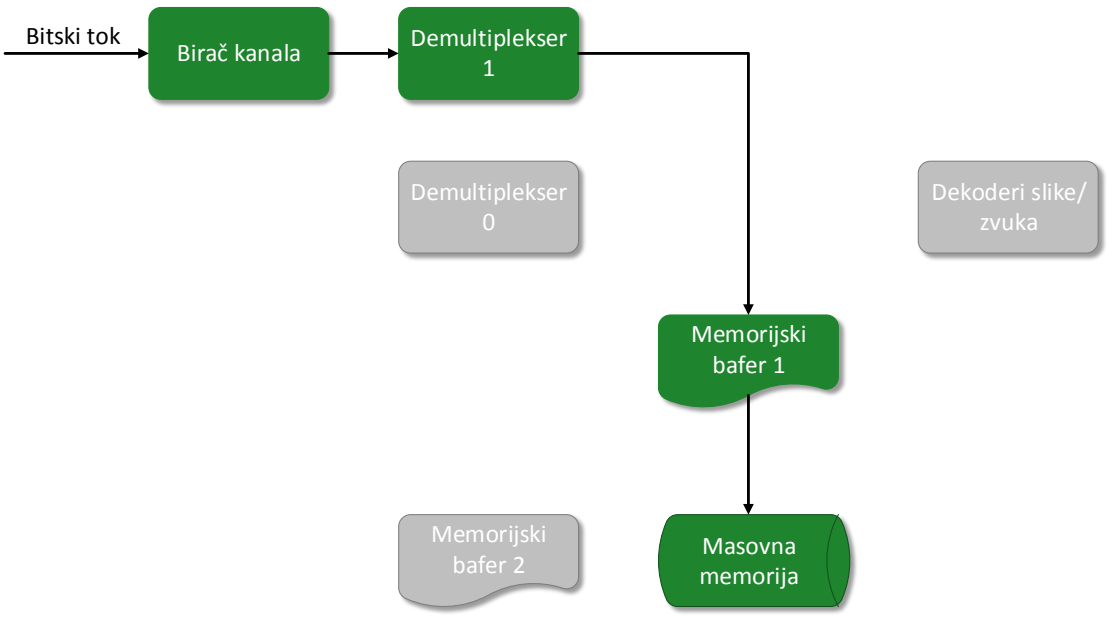
Sa slike se vidi da su za prikaz uživo emitovanog servisa potrebni elementi: birač kanala, demultiplekser i memorijski bafer. Koriste se po jedna instanca birača kanala i memorijskog bafera, dok je za ovaj slučaj potrebno kreirati dve instance demultipleksera. Iako je moguće dobiti prikaz servisa uz pomoć samo jedne instance demultipleksera, i bez korišćenja memorijskog bafera, razlog zbog koga se koristi struktura data na slici iznad je mogućnost nesmetanog istovremenog reprodukovanja i snimanja istog sadržaja u pozadini. Prilikom snimanja sadržaja u memorijski bafer, osim audio i video podataka, potrebno je snimiti i celokupan transportni tok vezan za taj TV servis, koji uključuje i dodatne propratne podatke kao što su prevod, ili teletekst, i stoga sadržaj koji završi u memorijskom baferu zapravo predstavlja jedan kompletan digitalni podkanal. Ovaj podkanal je po svojoj prirodi multipleksiran transportni tok izdvojen iz originalnog multipleksa transportnih tokova prisutnih na datoj frekvenciji, kojih može biti nekoliko u zavisnosti od stepena kompresije i rezolucije video sadržaja. Iz ovog razlog prva instanca demultipleksera (Demultiplekser 1 sa slike), služi za izdvajanje samo jednog digitalnog podkanala iz multipleksa digitalnih tokova (jedan TV servis), dok druga instanca demultipleksera (Demultiplekser 0 sa slike) služi za finalan prikaz podataka iz živog toka za prethodno izabran TV servis.

Konačan tok podataka u slučaju prikaza uživo emitovanog servisa je sledeći: Birač kanala vrši izbor fizičkog kanala koji sadrži željeni multipleks servisa i šalje TS (engl.

Transport **S**tream) pakete ka instanci demultipleksera sa indeksom 1. Tu se vrši prvo filtriranje paketa i u memorijski bafer šalju isključivo TS paketi koji u sebi sadrže komponente samo jednog servisa, odnosno PES (engl. **P**acketized **E**lementary **S**tream) podatke za taj servis (audio, video, prevod i teletekst). Potom memorijski bafer postaje izvor podataka i šalje ih ka instanci demultipleksera sa indeksom 0. Tu se vrši ponovno filtriranje TS paketa, ali sada samo onih koji u sebi sadrže komponente koje je korisnik izričito izabrao za prikaz, na primer, samo komponente zvuka i slike. Takođe, u ovom slučaju demultipleksiranja, vrši se i obrada filtriranih TS paketa, pa se ka dekoderima zvuka i slike šalju podaci u PES ili ES (engl. **E**lementary **S**tream) obliku.

5.2.3.5.2 Snimanje servisa

Da bi se sadržaj servisa snimio na neki masovni medijum, u određenoj formi je neizostavna akcija korisnika. Moguće je aktivirati snimanje na određeno dugme daljinskog upravljača, ili na primer snimanje koje će samo da počne u određeno zakazano vreme, zakazano preko događaja iz elektronskog programskog vodiča. Slika 5.12 ilustruje putanje podataka neophodne za osnovnu proceduru snimanja sadržaja, a druge akcije će, u kombinaciji sa snimanjem, biti objašnjene u nastavku.

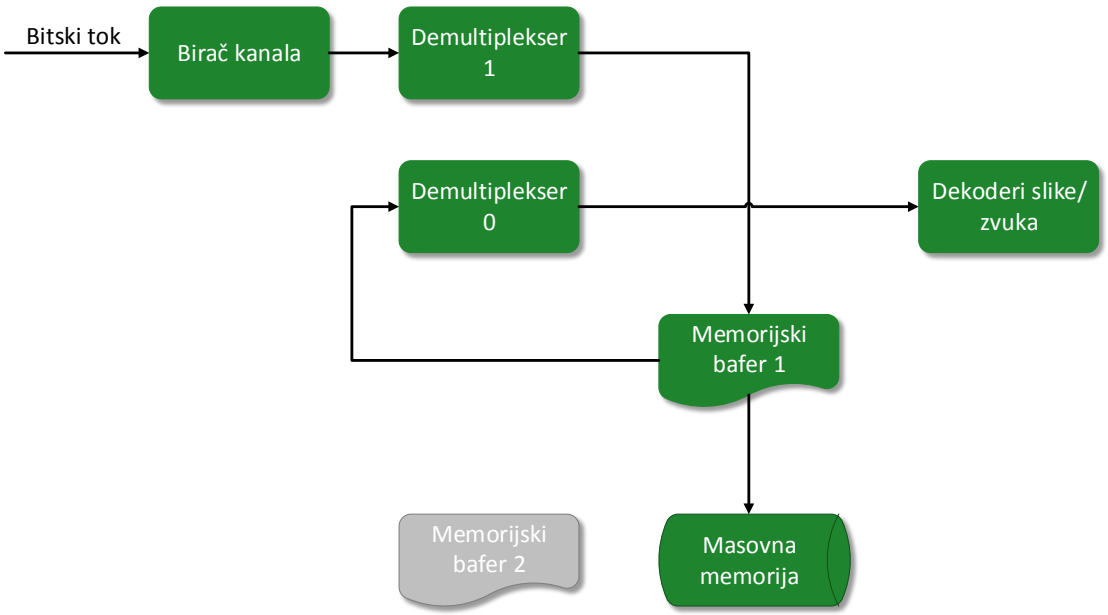


Slika 5.12 Prikaz putanje podataka prilikom snimanja servisa

Sa slike se vidi da su za snimanje servisa potrebni sledeći elementi: birač kanala, demultiplekser, memorijski bafer i masovni medijum. Za svaki od navedenih elemenata se koristi po jedna instanca. Kao i u prethodnom slučaju, birač kanala vrši izbor fizičkog kanala koji sadrži željeni servis. TS paketi se šalju ka instanci demultipleksera sa indeksom 1 gde se vrši filtriranje paketa i u memorijski bafer se šalju TS paketi izabranog TV servisa. Međutim, sada se TS paketi šalju ka masovnom medijumu gde se skladište za buduću upotrebu.

5.2.3.5.3 Istovremeno snimanje i prikaz uživo emitovanog servisa

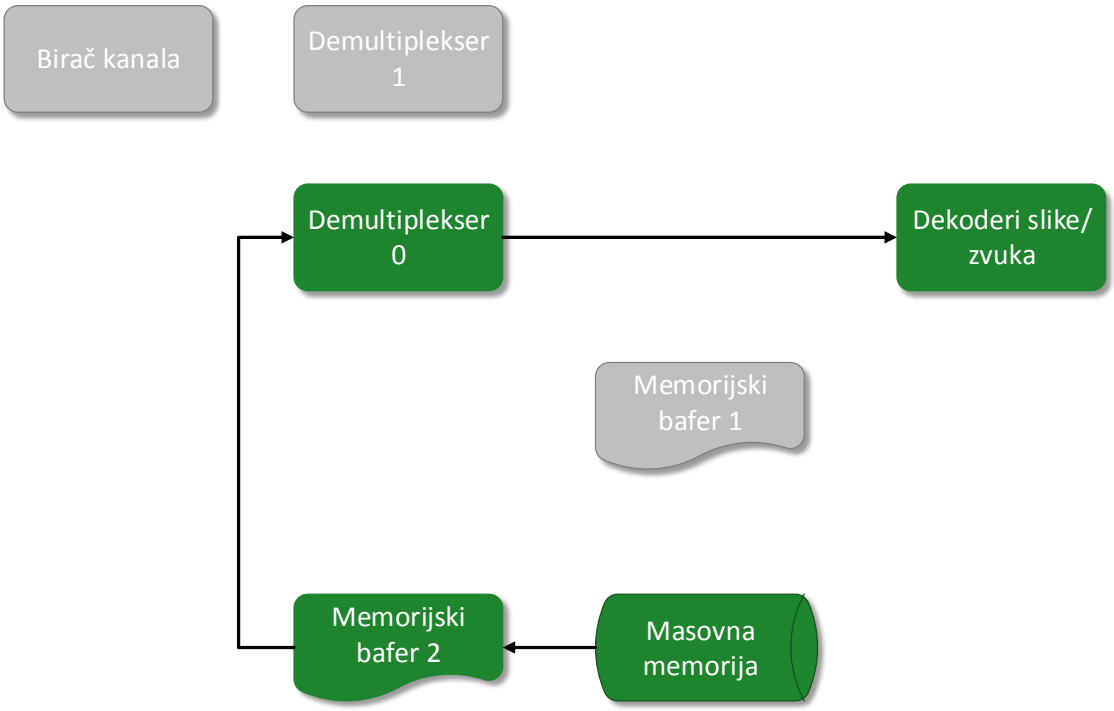
Putanja podataka koja je neophodna za proceduru snimanja i istovremene reprodukcije sadržaja prikazuje Slika 5.13. Za istovremeno snimanje i prikaz uživo emitovanog servisa, koriste se sledeći elementi: birač kanala, demultiplekser, memorijski bafer, masovni medijum i dekoderi zvuka i slike. Za svaki od navedenih elemenata se koristi po jedna instanca, osim za demultiplekser čije su dve instance neophodne za omogućavanje predviđene funkcionalnosti sistema. Birač kanala vrši izbor fizičkog kanala koji sadrži željeni servis. TS paketi se šalju ka instanci demultipleksera sa indeksom 1 gde se vrši filtriranje paketa koji pripadaju željenom TV servisu. Ti paketi se potom šalju u memorijski bafer. Za razliku od prethodna dva slučaja, TS paketi se iz memorijskog bafera šalju u dva pravca: ka masovnom medijumu (snimanje) i ka instanci demultipleksera sa indeksom 0, odnosno ka dekoderima zvuka i slike.



Slika 5.13 Prikaz putanje podataka prilikom istovremenog snimanja i gledanja servise uživo

5.2.3.5.4 Reprodukcija snimljenog servisa

Putanja podataka koja je neophodna za reprodukciju snimljenog servisa prikazuje Slika 5.14.



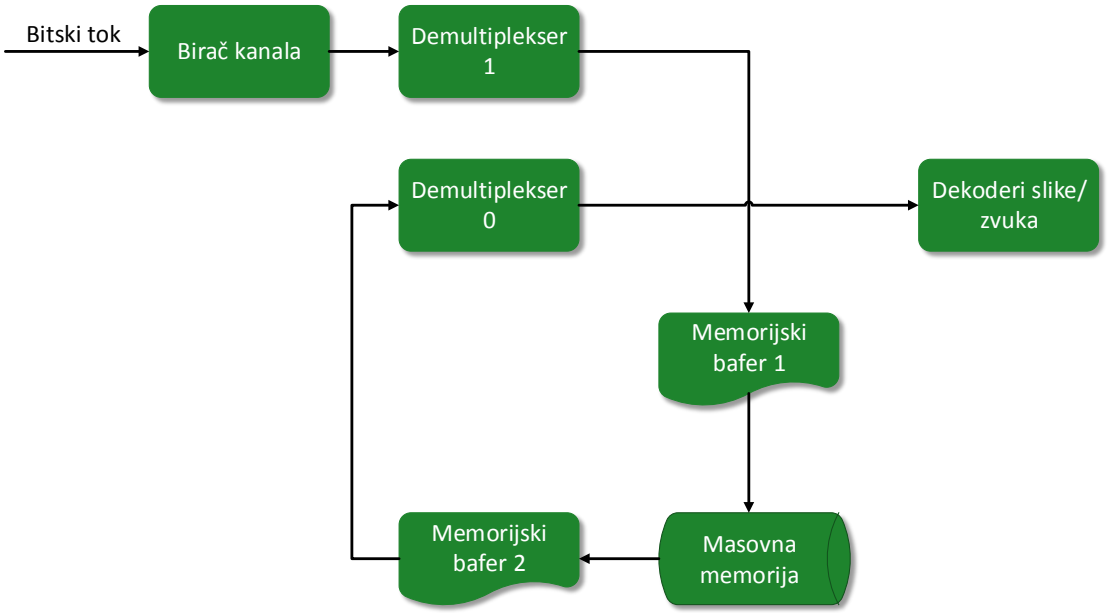
Slika 5.14 Prikaz putanje podataka korišćene prilikom reprodukcije snimljenog sadržaja

Za reprodukciju snimljenog servisa, koriste se sledeći elementi: masovni medijum, memorijski bafer, demultiplekser i dekoderi zvuka i slike. Za svaki od navedenih elemenata se koristi po jedna instanca. Sa masovnog medijuma se čitaju podaci koji su skladišteni u obliku TS paketa i šalju se ka memorijskom baferu. Odatle, TS paketi idu do demultipleksera gde se vrši njihovo filtriranje, a kasnije i obrada, slično kao u slučaju prikaza uživo emitovanog servisa. Nakon toga, podaci se u PES ili ES obliku šalju ka dekoderima zvuka i slike nakon čega se vrši prikaz snimljenog servisa.

5.2.3.5.5 Odloženo gledanje servisa

Putanja podataka koja je neophodna za odloženo gledanje servisa prikazuje Slika 5.15. Odloženo gledanje servisa čine sledeći elementi: birač kanala, demultiplekser, memorijski bafer, masovni medijum i dekoderi zvuka i slike. Po jedna instanca postoji za elemente: birač kanala, masovni medijum i dekoderi zvuka i slike. Po dve instance postoje za elemente: demultiplekser i memorijski bafer. Birač kanala vrši izbor fizičkog kanala

koji sadrži željeni servis. TS paketi se šalju ka instanci demultipleksera sa indeksom 1 gde se vrši filtriranje paketa koji pripadaju željenom TV servisu.



Slika 5.15 Prikaz putanje podataka korišćene prilikom odloženog gledanja servisa

Ti paketi se potom šalju u instancu memorijskog bafera sa indeksom 1. Nakon toga, paketi se skladište u masovnom medijumu. Istovremeno, vrši se čitanje tih paketa sa masovnog medijuma i oni se šalju ka instanci memorijskog bafera sa indeksom 2. Odatle paketi stižu do instance demultipleksera sa indeksom 0 nakon čega se na već opisan način podaci u PES ili ES obliku šalju ka dekoderima zvuka i slike.

5.2.3.6 Uslovni pristup/prikaz (CA/CI/CI+)

Predloženo Android4TV rešenje u sebi sadrži i podršku za uslovni prikaz TV servisa (CA/CI/CI+). Ova podrška uključuje parsiranje odgovarajućih DVB tabela (više detalja u dodatku "Servisne informacije (SI) DVB standarda") kao i propagaciju informacija o uslovnom prikazu kroz sve programske nivoe, zaključno sa odgovarajućim *DTVService* paketom (*android.dtv.ca*).

5.3 Sprega servisa digitalne televizije sa ostatkom Android sistema

Prethodno opisana neophodna proširenja Android OS i predložena programska sprega za integraciju servisa digitalne televizije nude podatke koji stižu iz DVB transportnog toka na nivou radnog okruženja za razvoj Android aplikacija. Predložena programska sprega omogućava razvoj TV aplikacija koje su u stanju da korisniku pruže standardne TV servise, kao i bilo koji drugi uređaj potrošačke elektronike koji nije baziran na Android OS.

Android OS je platforma otvorenog tipa i u sebi poseduje podršku za market aplikacija koji, u vreme pisanja disertacije (januar 2014. godine) broji preko 1000 000 različitih aplikacija [Ferrate], [Yaghmour] koje se mogu naknadno instalirati na uređaj. Usled nepostojanja standardizovane podrške za digitalne TV servise u okviru Android OS, ove aplikacije nisu u mogućnosti da iskoriste raspoložive informacije iz digitalnog TV transportnog toka prilikom svog izvršavanja.

Ovo predstavlja ozbiljnu manu prilikom izbora Android OS, kao ciljne platforme za TV/STB uređaje, jer korisnik nema mogućnost pristupa podacima iz DTV sveta kroz aplikacije koje su mu na raspolaganju. Programska sprega opisana u ovoj disertaciji pokušava da reši ovaj problem i pri tome nudi mogućnost potpune sprege podataka digitalnih TV servisa sa ostatkom Android ekosistema. Ovo daje mogućnost implementacije aplikacija koje do sada nisu bile moguće na TV platformama (i Android i baziranim na drugim OS) i koje mogu da transformišu način na koji korisnici vide jedan TV uređaj. U nastavku će biti predložene neke od potencijalnih sprega digitalne televizije u Android aplikacijama.

5.3.1 Pretraga TV baziranih podataka

Podaci od interesa za pretragu su podaci koji stižu u sistem putem DVB transportnog toka, i to putem PES paketa, odnosno PSI tabela (kao što je opisano u dodatku "Servisne informacije (SI) DVB standarda"). U cilju što boljeg sprezanja servisa digitalne televizije u Android OS ovo poglavlje analizira neophodna proširenja prethodno predložene programske sprege Androit4TV, da bi se omogućila pretraga DTV-baziranih podataka.

Na tržištu potrošačke elektronike iz domena TV prijemnika postoje rešenja, među kojima su neka rešenja i bazirana na Android OS (kao što je GoogleTV, [Ferrate]), koja pružaju korisniku pretragu podataka koji dolaze iz DTV domena. Iako ova rešenja imaju sve komponente fizičke arhitekture neophodne za pristup DVB transportnom toku (tjuner) ona i dalje DVB podatke (kao što je lista TV servisa, ili EPG događaji) obezbeđuju putem Interneta [Lukic]. Mana ovakvih rešenja je potencijalno pružanje zastarelih informacija korisniku, jer se informacije na Internetu, u praksi, mnogo ređe osvežavaju nego informacije koje stižu DVB transportnim tokom (gde se neke tabele osvežavaju na svakih nekoliko sekundi). Razlog za ovakav pristup je nepostojanje usaglašene i unificirane programske sprege za pristup DVB transportnom toku, odnosno servisima digitalne televizije u okviru Android operativnog sistema. Ovaj problem je moguće rešiti upotrebom predložene Android4TV programske sprege, uz izmene koje će biti opisane u narednim pasusima.

5.3.1.1 TV bazirani podaci od interesa za pretragu

Pre implementacije podrške za pretragu TV baziranih podataka, potrebno je definisati i odrediti koji podaci su od interesa za pretragu a da stiži iz DTV sveta. U ovoj implementaciji ograničili smo se na sledeća tri tipa podataka:

1. EPG događaji
2. Teletekst
3. PVR snimci

EPG (engl. **E**letronic **P**rogram **G**uide) je servis za televiziju, radio ili druge multimedijalne aplikacije koje prikazuju detaljnije informacije o sadržaju koji se trenutno emituje. Osnovna implementacija je opisana u standardima [ETSI2] i [ETSI3]. Ovi standardi definišu EPG podatke kao informacije koje se šalju putem tabela događaja (engl. **E**vent **I**nformation **T**able, EIT). EIT je deo PSI i SI podataka, kao što je definisano MPEG-2 i DVB standardima. EIT tabele su nezavisne za svaki servis, ali dele isti identifikator (PID) u okviru MPEG transportnog toka. Postoje dve različite grupe EIT tabela:

- **Trenutna/naredna**: Nosi informacije o emisiji (EPG događaju) koja se trenutno emituje i emisiji koje će se sledeća emitovati. Ove tabele se osvežavaju veoma često, najčešće na svake 2 sekunde.

- **Raspored**: Nosi informaciju o sadržaju (EPG događajima) koji će biti emitovan od momenta pristizanja tabele pa sve do 64 dana unapred. Frekvencija osvežavanja ovih tabela je značajno ređa, najčešće oko 30 sekundi.

Svaki EPG događaj je opisan sa nekoliko obaveznih polja, od kojih su, za pretragu, od interesa sledeća polja:

- **Kratak opis**: Ovo je osnovni opis događaja. Sadrži ime događaja i kratak opis. Ovo polje je maksimalne dužine od 255 bajta.
- **Produženi opis**: Ovo polje sadrži detaljan tekstualni opis EPG događaja. Kod emitera, ovo polje je poznato i pod imenom sinopsis. Maksimalna dužina predviđena standardom je 3984 bajta.

Teletekst, opisan standardom [ETSI1], je tip informacija koje se prosleđuju putem specijalnih PES paketa. Informacije su organizovane u strane alfanumeričkih simbola i jednostavnih grafičkih elemenata. Stranice se emituju kontinualno. Pristup samim stranicama iz TV aplikacije je moguć unosom trocifrenog broja koji predstavlja identifikator željene stranice, na daljinskom upravljaču. Teletekst stranica se sastoji od 23 vidljive linije, pri čemu svaka sadrži 40 karaktera. Svaki PES paket sadrži 45 bajtova, tako da se cela teletekst linija može preneti kao jedan paket. Jedan bajt predstavlja jedan karakter. Ostalih 5 bajtova se koriste za teletekst kontrolne informacije, kao što su jezik ili karakter-skup. Teletekst standard dozvoljava emitovanje 8 magazina, čiji identifikator predstavlja prvu cifru u trocifrenom broju koji na jedinstven način određuje jednu teletekst stranicu. U okviru svakog magazina teoretski može da se emituje do 256 stranica. Stoga druge dve cifre (u heksadecimalnom formatu) predstavljaju identifikator stranice u okviru jednog teletekst magazina. Npr. magazin 2 može sadržati stranice od broja 200 do 2FF.

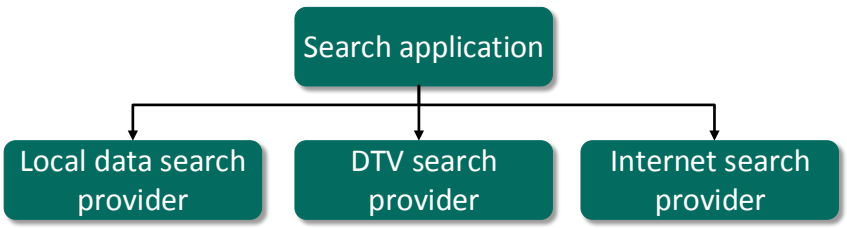
PVR, opisan standardom [ETSI7], pruža informacije od interesa za pretragu identične EPG događajima. Moderni TV uređaji i prateće PVR aplikacije ne snimaju samo digitalni video materijal, već i prateći EPG događaj koji opisuje TV servis koji se trenutno snima. Stoga PVR aplikaciju možemo gledati kao izvor identičnih informacija koje pruža i EPG aplikacija, uz dve razlike:

- PVR aplikacija dodaje dodatne informacije za pretragu kao što su dužina snimka

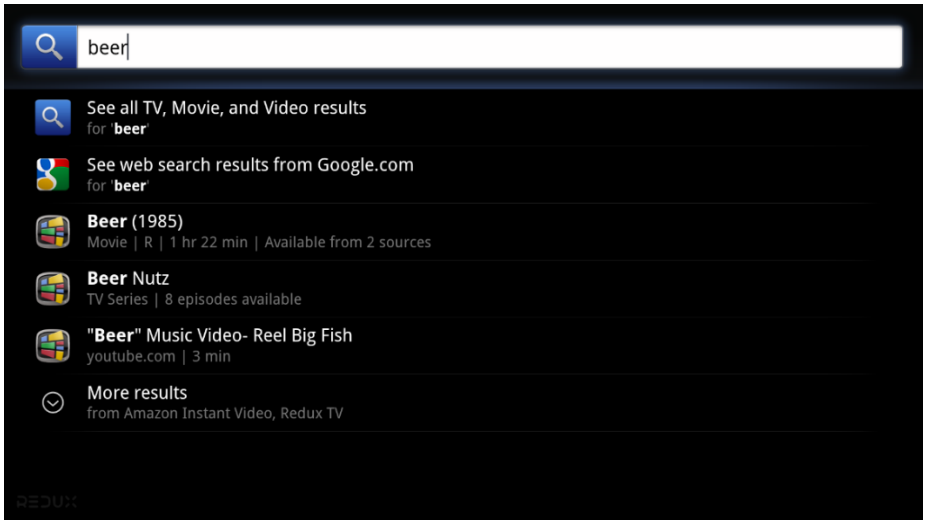
- EPG događaji u PVR aplikaciji ne dolaze direktno iz DVB transportnog toka, već iz masovnog medijuma gde je snimljen TV servis

5.3.1.2 Postojeća sistemska podrška u Android OS

Android operativni sistem u sebi sadrži sistemsku podršku za pretragu podataka korišćenjem unificirane sprege za pretragu. Ova unificirana pretraga se ogleda u postojanju globalne aplikacije koja vrši pretragu svih podataka u sistemu i pruža korisniku jedinstven GUI pristup za unos kriterijuma pretrage i prikaz rezultata pretrage. Aplikacija za pretragu (engl. **S**earch **A**pplication, SA) se oslanja na postojanje određenog broja dobavljača rezultata pretrage (engl. **S**earch **P**rovider, SP) u sistemu, koji SA obezbeđuju rezultate koji se prikazuju korisniku. SP moduli su zaduženi za stvarnu pretragu informacija, i to onog dela za koji su zaduženi. Jedan primer SP modula je modul za pretragu multimedijalnih datoteka u sistemu, čije je jedino zaduženje pretraga baze ovih datoteka sa prosleđenim kriterijumom pretrage. Svi SP moduli u sistemu se moraju prethodno registrovati od strane korisnika, da bi SA aplikacija mogla da im prosleđuje ključne reči za pretragu. Oni će po prijemu ključnih reči, pretražiti podatke za koje su zaduženi, i vratiti sve pronađene stavke SA aplikaciji u predefinisanom formatu. SA aplikacija po prijemu svih parcijalnih rezultata pretrage, formira konačnu listu, koja sadrži sve pronađene stavke na nivou sistema i prezentuje ih krajnjem korisniku. Da bi se ovaj mehanizam postigao, svi SP moduli implementiraju isti API za komunikaciju sa SA. Slika 5.16 prikazuje prethodno opisanu arhitekturu programske podrške za pretragu u okviru Android operativnog sistema. SA aplikacija u Android operativnom sistemu [Ableson], [Yaghmour] je poznata i pod imenom QSB (engl. **Q**uick **S**earch **B**ox). Slika 5.17 prikazuje izgled SA aplikacije na GoogleTV platformi [Ferrate].



Slika 5.16 Arhitektura programske podrške za pretragu podataka u Android OS

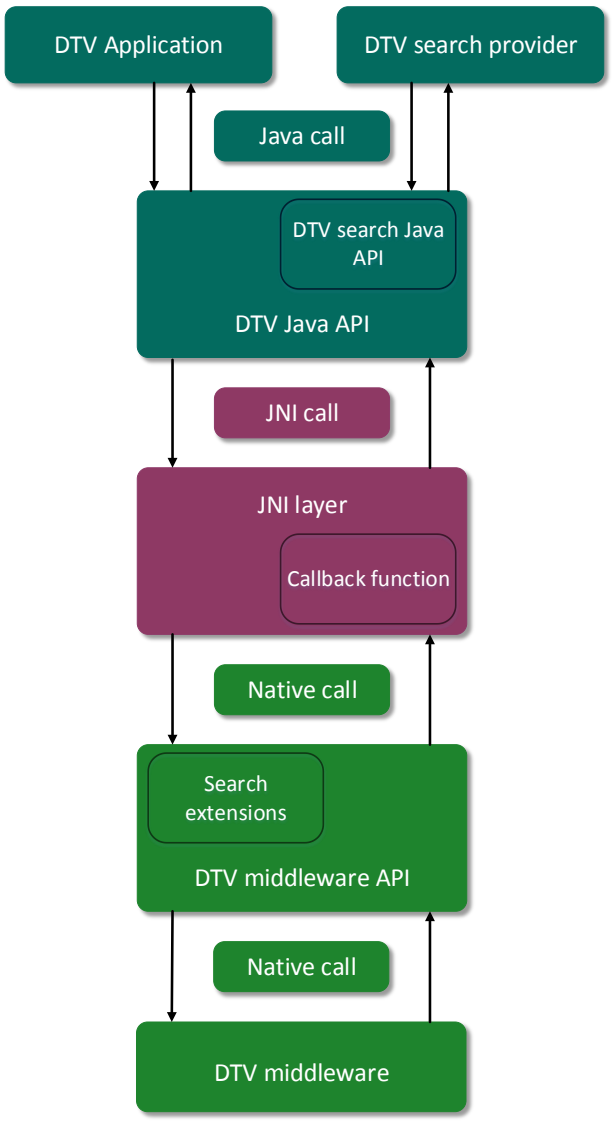


Slika 5.17 Izgled SA aplikacije na Google TV platformi

U spisku rezultata pretrage SA aplikacije na GoogleTV platformi, koji prikazuje Slika 5.17, osim rezultata iz standardnih SP modula (kakav je Internet pretraživač), nalaze se i rezultati koji dolaze iz TV domena. Ovi rezultati, kako je prethodno opisano, na GoogleTV platformi se obezbeđuju od strane namenskih servera (koje pruža i za čiju ažurnost se brine kompanija Google) putem Interneta. Sledeća poglavlja će demonstrirati kako je funkcionalnost pretrage DTV podataka koji dolaze iz DVB transportnog toka rešen korišćenjem predložene Android4TV programske sprege.

5.3.1.3 Dodatna programska sprega za pretragu podataka iz digitalnih TV servisa

S obzirom na to da je Java API za pretragu DTV sadržaja (engl. **DTV S**earch Java **API**, DTVSAPI) predstavlja proširenje predloženog programskog rešenja u poglavlju "Programska sprega za servise digitalne televizije", oslanja se na iste nivoe programske podrške. Pretraživi podaci iz DTV transportnog toka se prikupljaju i skladište u DTV MW modulu. DTVSAPI obezbeđuje te podatke višim programskim nivoima. Slika 5.18 prikazuje u više detalja kako DTV SP modul pristupa DTV MW podacima.

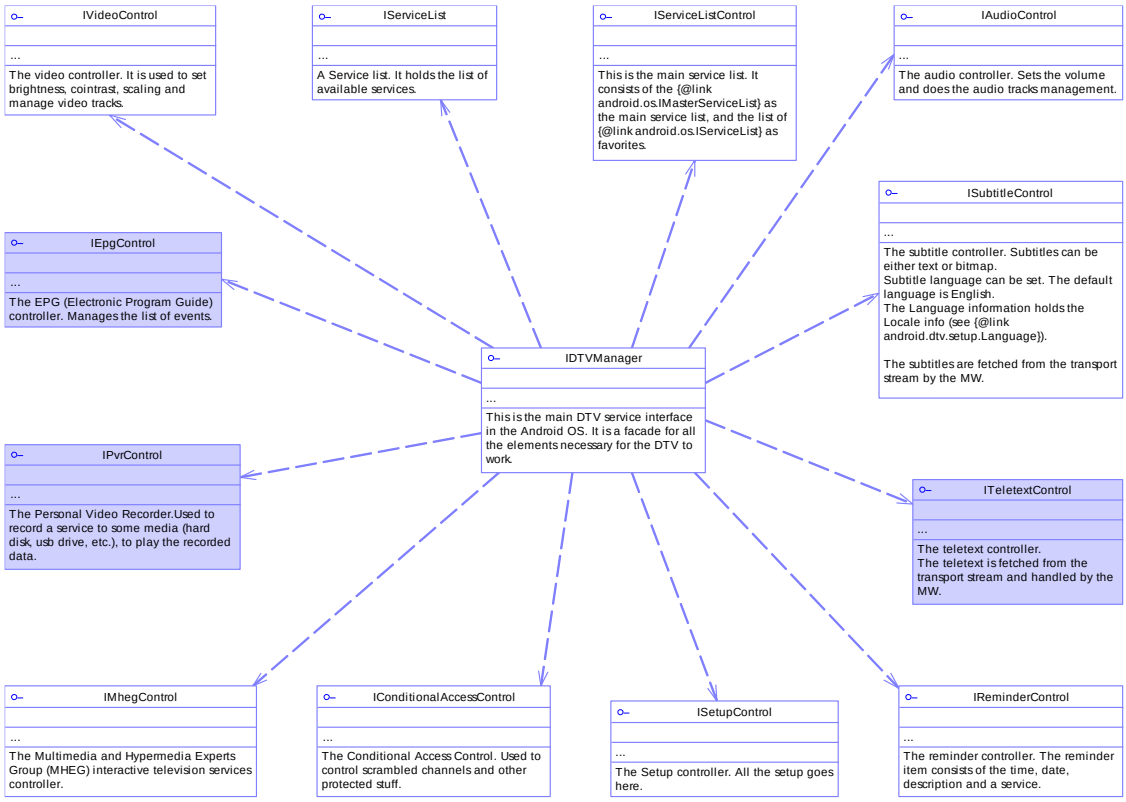


Slika 5.18 Komunikacija između DTV SP i DTV MW modula

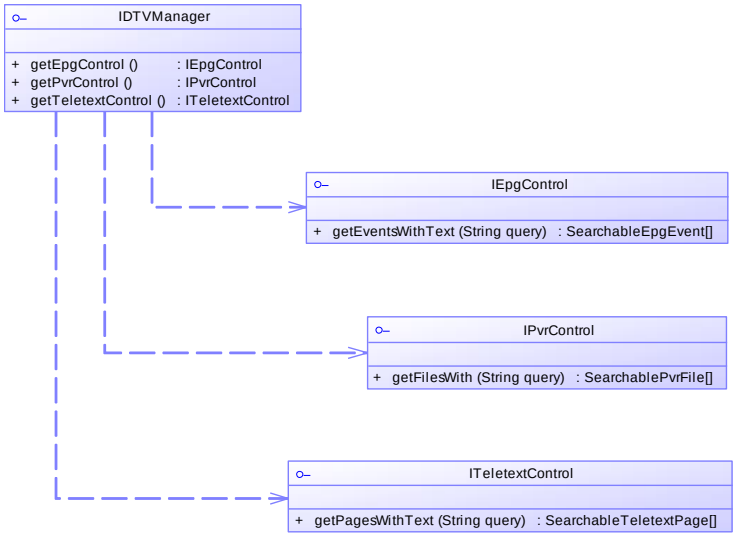
Komunikacija se oslanja na sledeće slojeve:

- DTV SP, koji je standardna Java Android aplikacija, registrovana od strane SA aplikacije.
- DTV Java API nivo, predloženo proširenje Android OS koje enkapsulira DTV servise u Java klase.
- JNI sloj, povezuje DTV Java API sa DTV MW modulom.
- DTV MW, izvršna programska podrška za digitalne TV servise.

Slika 5.19 pokazuje UML dijagrame klasa koje čine predloženi DTV Java API, sa označenim komponentama koje su od interesa za pretragu. Slika 5.20 prikazuje u više detalja proširene komponente.



Slika 5.19 UML dijagrami klasa koje čine predloženi DTV Java API (Android4TV)

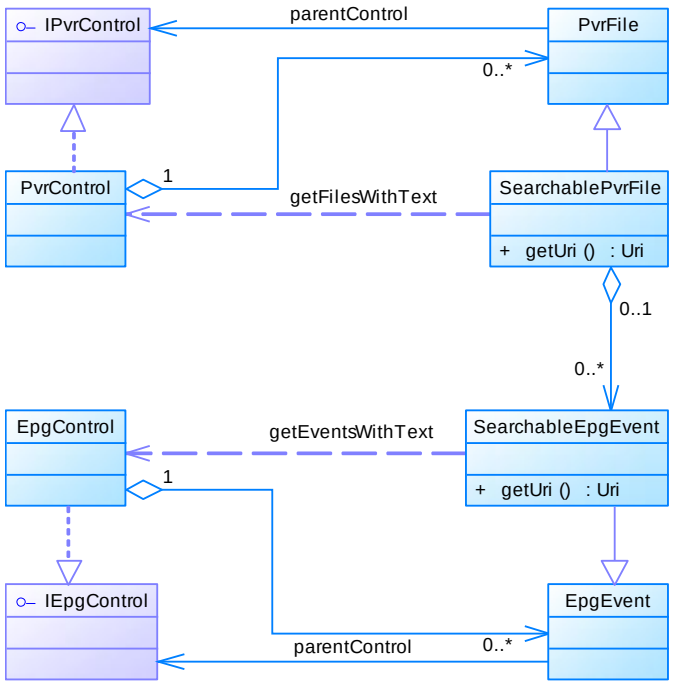


Slika 5.20 Proširene klase za pretragu

Svaki rezultat pretrage koji dolazi od svih SP modula registrovanih u sistemu, treba da obezbedi i akciju ukoliko se rezultat pretrage koji je dati SP modul obezbedio odabere od strane korisnika u SA. U slučaju DTV SP, ova akcija može biti jedna od sledećih:

- Za **EPG** rezultate: Ukoliko je rezultat emisija koja se trenutno prikazuje na nekom servisu digitalne televizije, TV aplikacija bi trebalo da se pokrene i pozicionira na taj servis. Ukoliko je EPG rezultat emisija koja je u budućnosti, tada je očekivano ponašanje da TV aplikacija prikaže dijalog koji će korisniku omogućiti da zakaže snimanje date emisije pomoću PVR funkcionalnosti.
- Za **teletekst** rezultate: Akcija treba da kaže TV aplikaciji da prikaže teletekst stranicu koja sadrži dati kriterijum pretrage.
- Za **PVR** rezultate: Akcija treba da pokrene prikaz prethodno snimljene TV emisije.

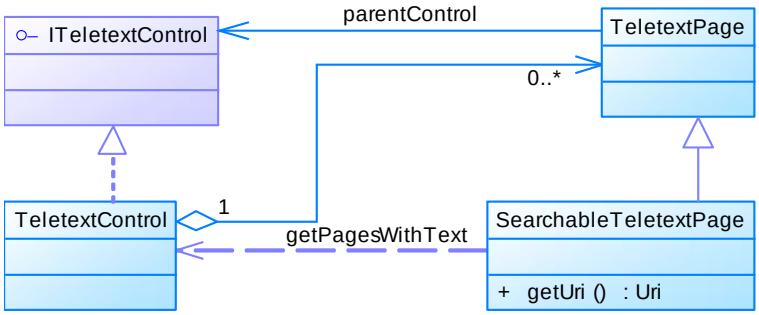
Da bi se postigla ova funkcionalnost, specijalno URI polje je dodato u klase koje sadrže DTV rezultate. Slika 5.21 prikazuje neophodna proširenja DTV Java API, uključujući proširene klase, neophodne za pretragu EPG i PVR podataka.



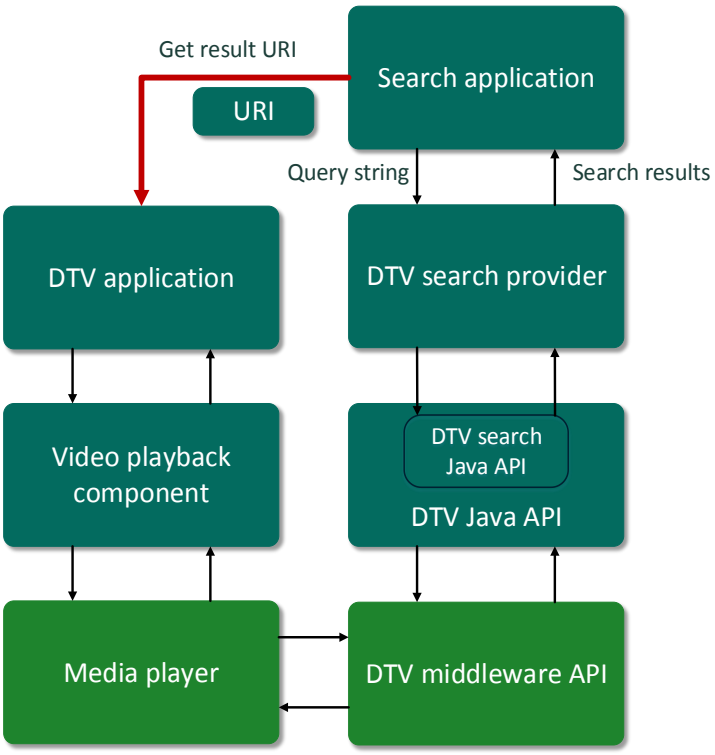
Slika 5.21 Neophodna proširenja za pretragu EPG i PVR podataka

Kada korisnik izabere DTV SP rezultat, SA će iskoristiti URI polje iz rezultata i proslediti ga TV aplikaciji, koja će potom izabrati odgovarajuću akciju. Slika 5.22 prikazuje neophodna proširenja za slučaj pretrage teletekst podataka.

Slika 5.23 prikazuje u više detalja kako ovaj mehanizam funkcioniše i sve slojeve programske podrške koji su neophodni da bi ovaj mehanizam pravilno funkcionisao.



Slika 5.22 Neophodna proširenja za pretragu teletekst podataka



Slika 5.23 Mehanizam izvršavanja akcije po odabiru DTV podataka iz SA aplikacije

5.3.1.3.1 Pretraga EPG sadržaja

EPG kontrolna sprega se koristi za pristup EPG listi događaja. Ova sprega je proširena operacijama za pretragu preko EPG tabele događaja.

```
IEpgControl epgCtl = dtvManager.getEpgControl();
```

EPG kontrolna sprega se koristi od strane DTV SP modula da bi se dobavili EPG rezultati pretrage. Svaki rezultat sadrži URI polje sa sledećim parametrima:

- Identifikator TV servisa
- Početno vreme EPG događaja
- Završetak EPG događaja
- Kriterijum pretrage

```
SearchableEpgEvent events[] = epgCtl.getEventsWithText(query);
for (SearchableEpgEvent event : events) {
    fillEpgSearchResult(event);
}
```

Kada se EPG rezultat izabere, URI polje rezultata se šalje TV aplikaciji, koja prvo proverava da li je EPG događaj u sadašnjosti ili budućnosti. Za sadašnje događaje, TV servis za koji je taj događaj vezan će biti prikazan, dok za buduće događaje, TV aplikacija ima mogućnost da prikaže korisniku dijalog koji sadrži opcije za zakazivanje PVR snimanja, ili podešavanje podsetnika.

```
Date eventDate = getUri().getStartingTimeParam().toDate();
if (eventDate <= nowDate) {
    videoPlayback.setVideoUri(
        service.toUri());
} else {
    showScheduleDialog(
        getUri().getServiceParam(),
        getUri().getStartingTimeParam(),
        getUri().getEndingTimeParam());
}
```

5.3.1.3.2 Pretraga teletekst sadržaja

Teletekst kontrolni modul se koristi od strane DTV SP za dobavljanje teletekst rezultata pretrage. Ova sprega je proširena operacijama za pretragu teletekst sadržaja.

```
ITeletextControl ttxCtl = dtvManager.getTeletextControl();
```

Teletekst kontrolna sprega se koristi u DTV SP da bi se izvršila akvizicija teletekst rezultata pretrage. Svaki rezultat sadrži URI polje sa sledećim parametrima:

- Identifikator TV servisa
- Broj teletekst strane
- Tekst pretrage

```
SearchableTeletextPage pages[] = ttxCtl.getPagesWithText(query);
for (SearchableTeletextPage page : pages) {
    fillTeletextSearchResult(page);
}
```

Kada korisnik izabere rezultat teletekst pretrage i odgovarajuće URI polje prosledi TV aplikaciji, identifikator servisa iz URI polja se koristi za dobavljanje i prikaz rezultujuće teletekst stranice.

```
ttxCtl.showTtxPage(getUri().getTeletextPageParam());
```

5.3.1.3.3 Pretraga PVR sadržaja

PVR kontrolna sprega se koristi za pristup operacijama za snimanje i odloženo gledanje TV servisa. Kako ova sprega ima pristup prethodno snimljenim datotekama, proširena je sa podrškom za pretragu.

```
IPvrControl pvrCtl = dtvManaget.getPvrControl();
```

PVR kontrolna sprega se koristi u DTV SP da bi se pribavili rezultati pretrage koji dolaze iz PVR domena.

```
SearchablePvrFile files[] = pvrCtl.getFilesWithText(query);
for (SearchablePvrFile file : files) {
    fillPvrSearchResult(file);
}
```

Svaki rezultat će sadržati URI polje sa sledećim parametrima:

- Deskriptor PVR video datoteke
- Tekst pretrage

Kada se PVR rezultat odabere i URI polje prosledi TV aplikaciji, prethodno snimljeni video materijal se može prikazati standardnim metodama:

```
videoPlayback.setDataSource(getUri().getPVRVideoUri());
```

5.3.1.4 Edit odstojanje

Tehnika poređenje stringova, poznata u literaturi kao "string-ka-stringu" korekcioni problem, važan je zadatak u mnogim oblastima razvoja programske podrške. *Edit* odstojanje, kao osnovni koncept za poređenje stringova, prethodno je bio predložen u literaturi [Levenshtein] i često se može naći i pod drugim imenom: Levenštajnovo odstojanje. Računanje *edit* odstojanja za dva stringa se najčešće računa upotrebom strategije dinamičkog programiranja (DP), koje se zasniva na razbijanju kompleksnih problema na manje celine. Ova strategija će biti opisana u narednom pasusu.

Ako je P konačni string dužine m nekog alfabeta Σ. U tom slučaju $P(i)$ je karakter na poziciji i u stringu P. $P(i, j)$ se u tom slučaju definiše kao podskup karaktera od pozicije i do pozicije j, iz stringa P. Prazan (null) string je predstavljen sa λ. *Edit* operacija je par $(p, x) \neq (\lambda, \lambda)$, takav da dužine stringova p i x su manje ili jednake vrednosti 1. Ova operacija se često navodi u literaturi kao $p \rightarrow x$. Tri *edit* operacije su:

1. ***Zamena***: Promena simbola p na poziciji i drugim simbolom $x \in \Sigma$ da bi se dobio string $p_1 \ldots p_{i-1}xp_{i+1} \ldots p_m$. Ova operacija se predstavlja sa $p \rightarrow x$.
2. ***Brisanje***: Brisanje simbola p na poziciji i da bi se dobio string $p_1 \ldots p_{i-1}p_{i+1} \ldots p_m$. Ova operacija se predstavlja sa $p \rightarrow \lambda$.
3. ***Umetanje***: Umetanje simbola $x \in \Sigma$ na poziciju i da bi se dobio string $p_1 \ldots p_ixp_{i+1} \ldots p_m$. Ova operacija se predstavlja sa $\lambda \rightarrow x$.

Funkcija cene γ je funkcija koja dodeljuje pozitivan realan broj svakoj *edit* operaciji $p \rightarrow x$, i obeležava se u literaturi sa $\gamma(p \rightarrow x)$. Cena izvršavanja skupa *edit* operacija se računa sabiranjem cena individualnih *edit* operacija. *Edit* odstojanje $\delta(P, X)$ transformacije stringa P u string X korišćenjem funkcije cene γ se definiše kao minimalna cena svih mogućih *edit* sekvenci transformacije stringa P u string X.

Standardni DP algoritam koji se može koristiti za utvrđivanje *edit* odstojanja između dva stringa je opisan u literaturi [Abe], [Bunke], odnosno [Wagner]. Algoritam funkcioniše po sledećem principu: Ako definišemo $D(i, j) = \delta(P(1, i), X(1, j))$, $0 \leq i \leq m$, $0 \leq j \leq n$. Tada, za sve $0 \leq i \leq m$, $0 \leq j \leq n$:

$$D(i,j) = \min \begin{cases} D(i-1, j-1) + \gamma(P(i) \to (X(j)), \\ D(i-1, j) + \gamma(P(i) \to \lambda) \\ D(i, j-1) + \gamma(\lambda \to X(j)). \end{cases}$$

Takođe, inicijalna vrednost za *D* se definiše na sledeći način:

$$D(0,0) = 0$$

$$D(0,i) = D(0,i-1) + \gamma(\lambda, X(i)), 1 \le i \le n$$

$$D(i,0) = D(i-1,0) + \gamma(P(i), \lambda), 1 \le i \le m$$

D(*i*, *j*) se može računati red po red, ili kolonu po kolonu prateći prethodne formule. Rešenje predstavlja minimalnu putanju koja počinje sa *D*(0, 0) i stiže u *D*(*m*, *n*). Očigledno ovaj algoritam ima složenost reda veličine *O*(*mn*).

5.3.1.4.1 Adaptacija algoritma računanja string odstojanja

U slučaju pretrage DTV podataka, važno je da se pretraga implementira na način koji je tolerantan na pravopisne greške i greške nastale prilikom samog unosa teksta pretrage (engl. spelling and typing errors). Glavna karakteristika ovog tipa grešaka je da unose mali (limitiran) broj devijacija u string koji sadrži kriterijum pretrage. Ovo znači da se može ograničiti maksimalno dozvoljeno odstojanje stringova koje se računa za metodu koja se koristi.

Ovaj proces se zove aproksimativno poređenje stringova, i ono dodaje još jedan parametar prethodnim jednačinama, *k*. Algoritam će prijaviti dva stringa da su jednaka ukoliko se ne razlikuju za više od *k* karaktera. Razlika može biti zamena, brisanje ili umetanje. U ovom slučaju, nije potrebno računati svaku *D*(*i*, *j*) vrednost. U [Ukkonen], predložena je izmena osnovnog algoritma koja smanjuje tabelu računanja. Ovo se postiže tako što se računaju samo vrednosti *D*(*i*, *j*) za koje je $|i - j| \le k$. Osnovna ideja je da ukoliko je $|i - j| > k$, tada je i $D(i, j) > k$, i nema potrebe za računanjem narednih vrednosti. Ovaj metod unapređuje kompleksnost na *O*(*kn*), pri čemu je jasna pretpostavka da je $k \le m$. Iako su u literaturi poznate tehnike koje postižu bolje performanse i dalje snižavaju kompleksnost računanja, kao što su tehnike predložene u [Baeza-yates]. Ove tehnike koriste posebne korake pripreme obrade, koje prethode samom mehanizmu poređenja stringova, pri čemu postižu bolje rezultate za velike skupove stringova koji se ne menjaju često, ako uopšte. Ova optimizacija omogućava

postizanje brzine obrade od preko 20 MB/s, prilikom pretraživanja velikih skupova stringova kao što su knjige i rečnici. Uzimajući u obzir da je za potrebe predložene implementacije količina podataka koju je potrebno pretražiti značajno manja, i menja se veoma često (teoretski podaci koji stižu putem DVB transportnog toka se mogu menjati svakih nekoliko sekundi), izabrana je metoda opisana u [Levenshtein].

Podaci od interesa za pretragu u DVB transportnom toku (kako EPG događaji tako i teletekst strane) mogu sadržati nekoliko reči koje sadrže kriterijum za pretragu. Pošto je od interesa pronaći makar jednu od tih reči, mehanizam pretrage se prekida onog momenta kada se pronađe prva reč koja je dovoljno slična kriterijumu za pretragu u podacima od interesa.

5.3.1.5 Implementacija

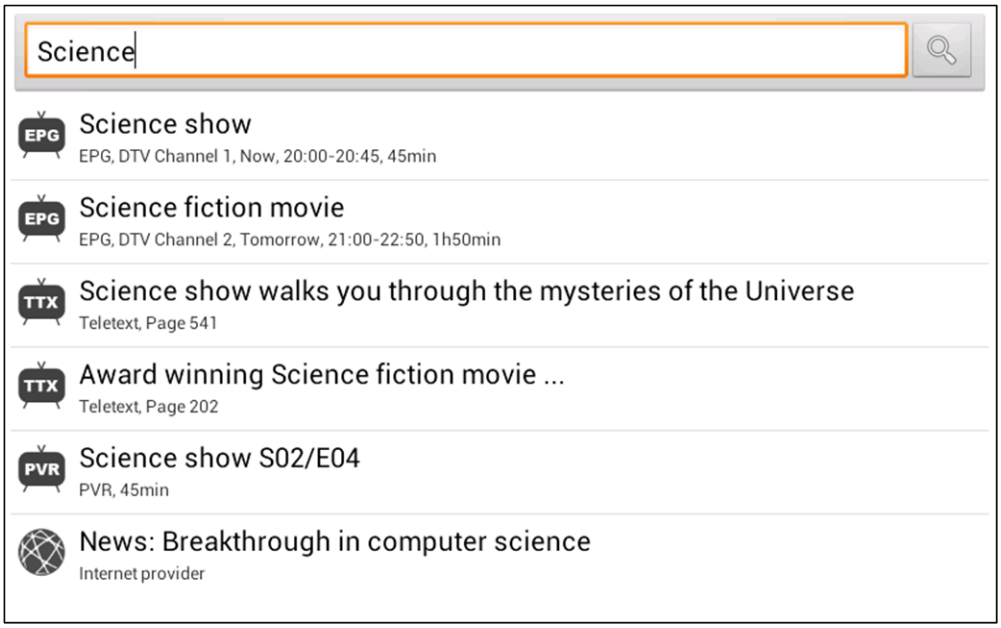
Aplikacija za pretragu DTV-baziranih podataka (DTV SP) je razvijena korišćenjem standardnih alata za razvoj Android aplikacija. Uzimajući u obzir da je cilj ove aplikacije da obezbedi DTV rezultate pretrage, ona implementira API za pretragu nametnut od strane Android OS. Ovaj API uključuje dve funkcionalnosti:

- Prijem kriterijuma pretrage od strane SA.
- Format rezultata pretrage koji se vraća nazad ka SA.

Kada se kriterijum pretrage unese u SA aplikaciju, onda ga dalje prosleđuje svim registrovanim SP aplikacijama u sistemu, uključujući i implementirani DTV SP. Po završetku pretrage DTV servisa, korišćenjem predloženih proširenja Android4TV programske sprege, DTV SP formatira rezultate u predefinisani format i prosleđuje ih SA aplikaciji koja je zadužena za prikaz krajnjem korisniku. Jedan primer DTV baziranih rezultata prikazuje Slika 5.24.

U ovom slučaju, DTV rezultati su u SA aplikaciji pomešani sa rezultatima koji su rezultat drugih SP u sistemu, kao što je Internet pretraživač. Takođe, važno je napomenuti da ova aplikacija (DTV SA) ne mora biti usko vezana za TV uređaje, već da se ona takođe može koristiti i na drugim uređajima baziranim na Android OS, kao što su telefoni i tableti, pod uslovom da se na njima može obezbediti DVB transportni tok (putem Interneta – IPTV, odnosno putem prenosivih USB DVB izvora). Na ovim uređajima, osim regularne pretrage, putem SA aplikacije, moguće je kombinovati DTV-baziranu pretragu i sa naprednim aplikacijama kao što su pretraga glasom, koja

je prirodno prisutna na telefonima/tabletima jer ovi uređaju uvek imaju integrisan mikrofon.



Slika 5.24 Primer izgleda SA aplikacije kada koristi rezultate pretrage od DTV SP modula

5.3.1.5.1 Ograničenja

Jedno značajno ograničenje Android OS predstavlja maksimalna količina podataka koja se može razmeniti između dva procesa prilikom jednog IPC prenosa. U pitanju je maksimalna vrednost jedne *Binder* transakcije, koja iznosi 1 MB. Uzimajući u obzir da su DTV SP i Java servis koji implementira digitalne TV servise (Android4TV) dva zasebna procesa, komunikacija između njih se mora realizovati upotrebom IPC mehanizma. U slučaju kada je kriterijum pretrage pronađen u svim teletekst stranicama, količina podataka koja u tom slučaju mora da se razmeni korišćenjem IPC poziva je 1.88 MB: Sličan problem se može javiti i prilikom EPG pretrage, gde u slučaju pogotka u svim EPG događajima, čija prosečna veličina je blizu 5 KB, na TV uređaju koji ima preko 100 TV servisa i svaki servis programski vodič za narednih 7 dana, limit od 1 MB po IPC prenosu lako može biti ugrožen. Postoji nekoliko različitih rešenja za ovaj problem:

1. Ograničiti broj rezultata pretrage DTV podataka.
2. Napraviti mehanizam DTV pretrage iterativnim.

3. Ograničiti dužinu kriterijuma pretrage DTV sadržaja.

Prvo rešenje ne zahteva izmene prethodno predloženog programskog rešenja, ali zato ni ne obezbeđuje sve rezultate pretrage korisniku. Ono se oslanja na dve pretpostavke: da neće biti toliko DTV podataka pronađenih prilikom pretrage (Tabela 5.1 prikazuje maksimalan broj DTV podataka, podeljenih po tipu, u jednom IPC prenosu), odnosno da i ukoliko ima toliko rezultata da oni prevazilaze mogućnosti IPC prenosa, tada to indicira da sam kriterijum pretrage nije dovoljno dobro konstruisan (ili je previse opšti). Ovaj pristup je direktno povezan sa trećim predloženim rešenjem.

Tip DVB podataka	Veličina jednog rezultata pretrage	Prosečan broj podataka (za jedan servis)	Maksimalan broj podataka u jednom IPC prenosu
EPG	5 KB	168	204
Teletekst	920 B	900	1200
PVR	5 KB	/	204

Tabela 5.1 Količina podataka potreba za prenos svakog tipa podataka. Maksimalan broj podataka u okviru jednog IPC prenosa.

Drugo rešenje dozvoljava da se svi rezultati pretrage proslede ka krajnjem korisniku, bez ograničenja, ali zato su potrebne određene izmene u prethodno predloženoj programskoj sprezi za pretragu. Ove izmene se ogledaju u dodavanju posebnih parametara svakoj funkciji za dobavljanje DTV rezultata, koji će govoriti koji raspon rezultata se želi dobaviti. Osim ovog proširenja, potrebno je i dodati posebne funkcije za svaki tip DTV rezultata koje će se koristiti da se utvrdi ukupan broj rezultata.

Treće rešenje ograničava minimalan broj karaktera u kriterijumu pretrage za koji će DTV SP vratiti rezultate. Ovo značajno redukuje broj rezultata koji se mogu pojaviti. Postavljanje ove vrednosti na 3, u izvedenim eksperimentima sa TV prijemnicima i standardnim DVB-T transportnim tokovima, se pokazalo kao dovoljno da se količina rezultata dovoljno redukuje da stane u jedan IPC prenos.

Tabela 5.2 prikazuje kako dužina kriterijuma pretrage (SC) utiče na broj pronađenih rezultata u odnosu na ukupan broj podataka koje je moguće pretražiti.

Tip DVB podataka	SC = 1 karakter	SC = 2 karaktera	SC = 3 karaktera
EPG	98%	64%	47%
Teletekst	95%	67%	24%
PVR	93%	65%	40%

Tabela 5.2 Uticaj dužine kriterijuma pretrage (SC) na broj pronađenih rezultata u odnosu na ukupan broj podataka koje je moguće pretražiti

Eksperimenti takođe pokazuju da nije dovoljno samo ograničiti minimalan broj karaktera u kriterijumu pretrage. Potrebno je takođe definisati način pretraživanja DTV podatke za kriterijume pretrage koji su duži od 3. Korišćenjem samo tradicionalnih mehanizama za traženje manjeg stringa u većem (koji pretražuje veći string i traži deo koji je identičan kraćem stringu) čini pretragu netolerantnom sa stanovišta pravopisnih i grešaka prilikom kucanja. Sa druge strane, korišćenje algoritma predloženog u poglavlju "Adaptacija algoritma računanja string odstojanja" će rešiti ovaj problem, ali će u rezultatima pretrage zanemariti sve stringove koji se razlikuju po dužini za *k* karaktera od kriterijuma pretrage. Da bi se rešila oba problema, predložen je sledeći postupak [Lukic]:

- Mehanizam pretrage za DTV podatke će vraćati rezultate samo ukoliko je kriterijum pretrage duži od 3 karaktera. U tom slučaju, tradicionalna tehnika za pronalaženje podstringa će se koristiti.
- Ukoliko je kriterijum duži od 4 karaktera, tada se obe tehnike (podstring i modifikovano Levenshtein odstojanje) koriste odvojeno, a njihovi rezultati kombinuju za prikaz krajnjem korisniku.

Za računanje Levenshtein odstojanja, koristi se aproksimativno poređenje stringova. Određivanje odgovarajuće vrednosti za parametar *k* može takođe da utiče na kvalitet i broj pronađenih rezultata pretrage. Za kratke kriterijume pretrage, koji se sastoje od nekoliko karaktera, postavljanje *k* parametra na vrednost koja je uporediva sa dužinom kriterijuma pretrage rezultovaće pronađenim DTV podacima koji nemaju sličnosti sa polaznim kriterijumom za pretragu. Ovo je dovelo do zaključka da *k* parametar mora da bude u vezi sa dužinom kriterijuma za pretragu. U eksperimentima, najbolji rezultati su postizani kada je *k* postavljano na 25% vrednosti dužine kriterijuma pretrage.

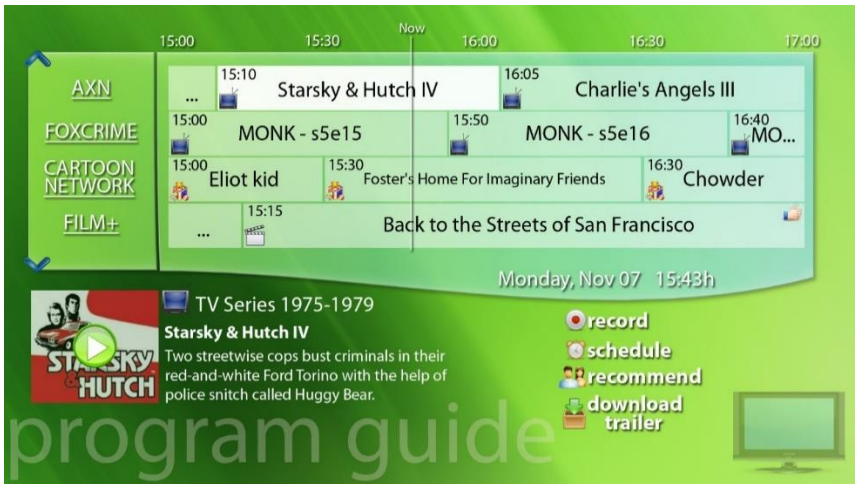
5.3.2 Predlog drugih potencijalnih tipova sprega

5.3.2.1 Napredni TV vodič

Sa prethodno opisanom programskom spregom, Android aplikacije su u mogućnosti da dobave EPG podatke. Takođe, sve postojeće socijalne mreže obezbeđuju razvojne programske sprege za programere koji žele da razvijaju Android aplikacije bazirane na tim socijalnim mrežama. Spajanjem ova dva sveta, moguće je razviti TVC aplikaciju za prikaz EPG događaja, ali koja uz standardne i očekivane informacije o raspoloživim emisijama pruža i mogućnost da:

- Informaciju koju emisiju korisnik trenutno gleda podeli sa svojim prijateljima na društvenoj mreži.
- Korisnik oceni odgledanu emisiju.
- Korisnik napiše svoj komentar vezan za odgledanu emisiju.
- Dobije preporuku emisija koju su njegovi prijatelji na društvenim mrežama označili sa pozitivnom ocenom.

Ovakva sinergija pruža mogućnost daljeg unapređenja aplikacija za TV prijemnike, korišćenjem otvorenog razvojnog okruženja Android operativnog sistema, koja ima potencijal da transformiše način na koji korisnici koriste TV uređaj, odnosno u ovom slučaju programski vodič. Slika 5.25 prikazuje primer jedne implementacije naprednog TV programskog vodiča.



Slika 5.25 Primer jedne implementacije naprednog TV programskog vodiča koji koristi mogućnosti socijalnih mreža

5.4 Eksperimentalna potvrda predloženog programskog rešenja

Za potrebe eksperimentalne potvrde predložene programska sprege i neophodnih proširenja Android operativnog sistema za servise digitalne televizije izabrano je nekoliko platformi. Prilikom odabira platformi za potvrdu, uzete su u obzir sledeće karakteristike predloženog rešenja u ovoj disertaciji:

1. Platformska nezavisnost
2. Kompletnost
3. Obrada u realnom vremenu
4. Nezavisnost od verzije Android operativnog sistema

Predloženo rešenje rešava problem proširenja Android operativnog sistema servisima digitalne televizije koje je nezavisno od platforme na kojoj se izvršava. Platformska nezavisnost se može posmatrati na dva nivoa:

- *Platformska nezavisnost iz ugla korisnika programske sprege (TV aplikacija)*: Ovaj aspekt je garantovan činjenicom da je programska sprega pisana Java programskim jezikom i pozicionirana na nivou Android aplikativnog radnog okruženja.
- *Platformska nezavisnost iz ugla integrisanog kola na kojem se izvršava*: Ovaj aspekt je od izuzetne važnosti jer se vremenski kritičan deo obrade predloženog rešenja nalazi na izvršnom nivou programske podrške Android operativnog sistema (pisan u C/C++ programskom jeziku).

Da bi se eksperimentalno potvrdilo rešenje sa stanovišta platformske nezavisnosti na oba prethodno opisana nivoa, odabrane su platforme koje se razlikuju po:

- *Tipu procesorske jedinice koja se koristi*: Današnje tržište digitalnih TV prijemnika se oslanja na dva tipa procesorskih jedinica – MIPS i ARM. Oba tipa se nalaze prisutna u odabranim platformama.
- *Broju procesorskih jezgara*: Integrisana kola u modernim TV prijemnicima prate trend povećanja broja procesorskih jezgara koji je vidljiv na tržištu personalnih računara. Odabrane platforme poseduju procesore koji sadrže jedan, dva i četiri procesorska jezgra, što ujedno i verno preslikava trenutne trendove u potrošačkoj elektronici.

- *Količini raspoložive memorije*: Količina radne memorije ima, zajedno sa procesnom moći procesorske jedinice, najveći udeo prilikom definisanja mogućnosti ciljne platforme. Odabrane platforme se značajno razlikuju po količini memorije raspoložive programskoj podršci.

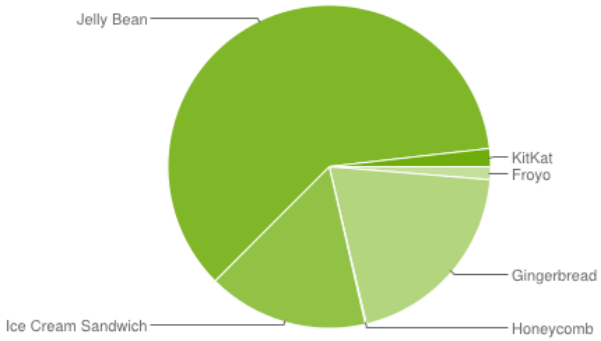
Osim platformske nezavisnosti, rešenje treba da garantuje kompletnost u smislu količine podržanih servisa digitalne televizije. Iz tog razloga, platforme su birane tako da podržavaju, na nivou fizičke arhitekture, sve sprege neophodne za moderne digitalne TV servise. Neke od neophodnih sprega fizičke arhitekture:

- *Tjuner*: DVB-T/T2/C/C2/S/S2
- *Modul za uslovni pristup*: CAM (engl. **C**onditional **A**ccess **M**odule)
- *Ulazne video sprege*: HDMI, CVBS, S-Video, SCART, VGA

Važna karakteristika predloženog rešenja je i nezavisnost od verzije Android operativnog sistema. Ova karakteristika garantuje da će predloženo rešenje biti u mogućnosti da i u budućnosti, sa novijim verzijama operativnog sistema i dalje bude konkurentno i primenljivo. Uzimajući u obzir da je dinamika promena verzija Android operativnog sistema u skladu sa brzo-promenljivim tržištem potrošačke elektronike, za potvrdu rešenja su odabrane verzije koje trenutno čine većinski udeo na tržištu uređaja koji poseduju Android OS. Tabela 5.3 i Slika 5.26 prikazuju trenutne trendove (februar 2014), kada je u pitanju zastupljenost različitih verzija Android operativnog sistema na tržištu potrošačke elektronike.

Verzija	Kodno ime	API nivo	Distribucija
2.2	Froyo	8	1.3%
2.3.3 - 2.3.7	Gingerbread	10	20.0%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	16.1%
4.1.x	Jelly Bean	16	35.5%
4.2.x		17	16.3%
4.3		18	8.9%
4.4	KitKat	19	1.8%

Tabela 5.3 Zastupljenost različitih verzija Android OS na tržištu potrošačke elektronike



Slika 5.26 Dijagram zastupljenosti različitih verzija Android OS

Za kraj, Tabela 5.4 prikazuje odabrane platforme za eksperimentalnu potvrdu predloženog rešenja. Platforme su odabrane tako da ispune prethodno opisane kriterijume, a i da u isto vreme budu konkurentne sa industrijskim rešenjima prisutnim na tržištu. Izabrane su sledeće platforme:

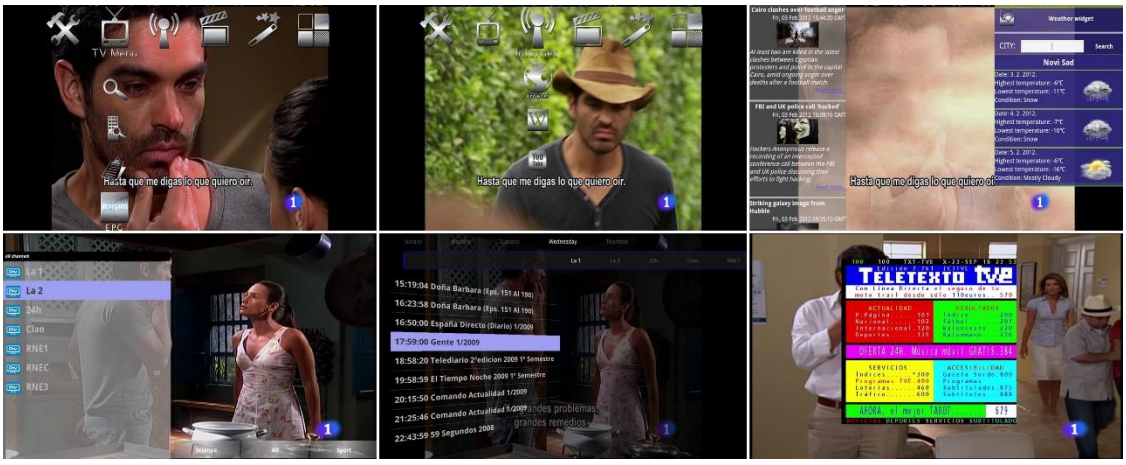
- [Armada 1500 Pro] kompanije Marvell
- [BCM7425] kompanije Broadcom
- [MPQ8064] kompanije Qualcomm
- [STiH 416] kompanije STMicroelectronics

Za potrebe eksperimentalne potvrde rešenja na odabranim platformama, razvijena je grafička programska podrška za TV prijemnik. Ova TV aplikacija služi za testiranje kako funkcionalnosti implementiranog rešenja, tako i potvrdu performansi koje se ogledaju u obradi u realnom vremenu. S obzirom na to da je predloženo programsko rešenje u disertaciji zapravo Java programska sprega, ista TV aplikacija se koristila na svim platformama za testiranje. Time su dokazana platformska nezavisnost i univerzalnost rešenja, što je ujedno i bio glavni cilj predloženog rešenja.

Slika 5.27 prikazuje korisničku spregu realizovane TV aplikacije za eksperimentalnu potvrdu. TV aplikacija u sebi poseduje svu funkcionalnost, koju nabraja Tabela 4.1. S obzirom na to da sve korišćene platforme ne poseduju jednake karakteristike fizičke arhitekture, kao što su tip tjunera, podrška za komponente za uslovni pristup (CAM), neke od funkcija u okviru TV aplikacije nisu bile funkcionalne na platformama koje ne poseduju podršku za željene karakteristike sa stanovišta fizičke arhitekture.

Platforma / Karakteristika	Armada 1500 Pro	MPQ8064	BCM7425	STiH 416
SoC	Marvell	Qualcomm	Broadcom	STM[1]
CPU	ARM A9	ARM v7	MIPS32	ARM A9
Takt CPU	1.2 GHz	1.5 GHz	1.3 GHz	1.2 GHz
Broj CPU jezgara	4	2	1[2]	2
CPU arhitektura	ARM	ARM	MIPS	ARM
RAM	2 GB	2 GB	512 MB	1 GB
GPU	Vivante GC4000	Adreno 320	SoC[3]	Mali 400-MP
Tjuner	DVB-T	DVB-T/T2/C	MoCA	DVB-S
Verzija Android OS	4.2.2	4.4	4.0.3	4.0.3
NAPOMENE:		[1] STMicroelectronics [2] Superskalarna arhitektura koja je u mogućnosti da izvršava dve programske niti u paraleli [3] Broadcom zaštićeno GPU rešenje		

Tabela 5.4 Odabrane platforme za potrebe eksperimentalne potvrde predložene programske podrške



Slika 5.27 Izgled korisničke sprege realizovane TV aplikacije za eksperimentalnu potvrdu predloženog rešenja

POGLAVLJE 6.

REZULTATI MERENJA

6.1 Ocena kvaliteta rešenja

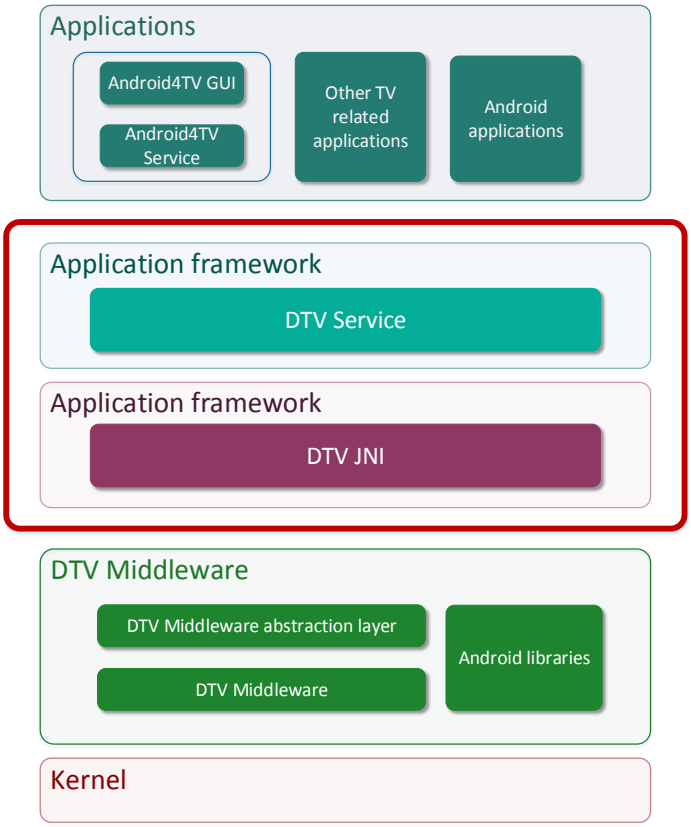
Upotreba metrika da bi se kontrolisao, predvideo i unapredio kvalitet proizvoda koji je baziran na programskoj podršci u poslednje vreme dobija na značaju i popularnosti. Postoji veliki izbor različitih skupova metrika koje se mogu koristiti u toku životnog veka projekta, ali uzimajući u obzir domen predložene programske podrške koja je opisana u ovoj disertaciji, akcenat je dat na objektno orijentisane (OO) metrike za merenje kvaliteta programske podrške. Kao što je predloženo u poglavlju "Mere za ocenu kvaliteta objektno orijentisane programske sprege", analiza je izvršena na četiri različita nivoa:

1. Sistemski nivo
2. Nivo nasleđivanja
3. Nivo klasa
4. Nivo metoda klasa

Pre početka same analize, potrebno je bilo utvrditi skup modula koji će biti analizirani. Uzimajući u obzir predmet istraživanja opisan u ovoj disertaciji, potom doprinos i cilj predloženog programskog koncepta pod imenom Android4TV, logičan izbor je bio da se analizira sloj Java programske sprege koji obezbeđuje DTV funkcionalnost. Koncept Android4TV obuhvata sledeći skup ciljeva:

- Definisanje skupa pravila za proširenje jezgra Android operativnog sistema, tako da se omoguće servisi digitalne televizije.
- Definisanje i implementacija platformski nezavisne programske podrške za servise digitalne televizije, koja uključuje:
    - Java baziranu programsku spregu, pozicioniranu u Android razvojnom aplikativnom sloju.
    - Android TV aplikaciju koja se oslanja na prethodno definisanu programsku spregu i implementira standardne servise digitalne televizije.

Iz ovoga možemo zaključiti da se jezgro doprinosa ove disertacije ogleda u programskoj sprezi baziranoj na Java programskom jeziku, čiji kvalitet je od velike važnosti prilikom analize i ocene kvaliteta predloženog rešenja. Slika 6.1 prikazuje module od interesa za dalju analizu u ovom poglavlju, uokvirene crvenom bojom i opisane u više detalja u poglavlju "Programska sprega za servise digitalne televizije".



Slika 6.1 Programski moduli od interesa za analizu predložene programske podrške

6.1.1 Ocena kvaliteta rešenja na sistemskom nivou

Metrike na sistemskom nivou se mogu izvesti iz metrika na nivou klasa uz dodavanje odgovarajuće statistike, čineći ih tako relativnim merama koje identifikuju sisteme programske podrške koji odstupaju od norme. Za ovaj nivo merenja, korišćen je skup metrika pod imenom MOOD, opisan u [Abreu1]. Ovaj skup metrika analizira sledeće mehanizme objektno orijentisanog programiranja:

- enkapsulacija (MHF i AHF)
- nasleđivanje (MIF i AIF)
- polimorfizam (POF)
- propuštanje poruka (COF)

Prilikom merenja *DTVService* i *JNI* slojeva programske podrške MOOD skupom metrika, dobijeni su sledeći rezultati, koje prikazuje Tabela 6.1:

Metrika	Vrednost
MHF	1.03%
AHF	100%
MIF	49.66%
AIF	0%
POF	Nedefinisan
COF	1.06%

Tabela 6.1 Rezultati merenja MOOD skupom metrika

Zaključci o kvalitetu analizirane programske podrške se ne mogu izvesti samo na osnovu izmerenih vrednosti, jer se ove vrednosti moraju uporediti sa usaglašenim normama kada je MOOD metrika u pitanju. Postoji veliki broj radova koji se bave temom prihvatljivih pragova svake od metrika iz MOOD skupa. U radovima [Abreu1], [Abreu2], [Abreu3], [Abreu4], [Harrison2] i [Al-Ja'afer1] predloženi su minimalni i maksimalni dozvoljeni pragovi za svaku od metrika. Ukoliko programska podrška koja se analizira zadovoljava predloženi raspon pragova, može se smatrati da ispunjava

uslove kvaliteta sa stanovišta objektno orijentisane paradigme, na sistemskom nivou. Tabela 6.2 prikazuje ove pragove.

Metrika	Izmerena vrednost	Minimalna preporučena vrednost metrike	Maksimalna dozvoljena vrednost metrike
MHF	1.03%	12.7%	21.8%
AHF	100%	75.2%	100%
MIF	49.66%	66.4%	78.5%
AIF	0%	52.7%	66.3%
POF	Nedefinisan	2.7%	9.6%
COF	1.06%	0%	11.2%

Tabela 6.2 Maksimalne dozvoljene vrednosti metrika MOOD skupa predložene u literaturi

Iz rezultata koje prikazuje Tabela 6.2, vidi se da određene metrike ukazuju na potencijalne nedostatke programske podrške. Ovi izmereni potencijalni nedostaci se mogu opravdati strukturom i svrhom programske podrške koja je razvijena i testirana. MOOD skup metrika se trudi da na objektivan način proceni kvalitet programske podrške, ali to u opštem slučaju nije moguće uraditi na univerzalan način i bez uzimanja u obzir arhitekture i svrhe programskog koda koji se analizira.

MHF metrika meri nivo skrivanja metoda u okviru klasa u sistemu programske podrške, kao mere jednog od osnovnih principa OO dizajna – enkapsulacije. U predloženom rešenju, programska podrška koja se analizira je programska sprega (API) i kao takva, očekivano je da sadrži veliki broj otvorenih metoda (tipa *public*) koje su na raspolaganju višim programskim nivoima. Osim što je u pitanju programska sprega, analizirani sistem je namenski pisan da ima plitko stablo nasleđivanja, o čemu će biti detaljnije diskutovano u nastavku.

MIF metrika meri nivo nasleđivanja metoda u okviru klasa u sistemu programske podrške, kao mere jednog od osnovnih principa OO dizajna – nasleđivanja. Plitko stablo nasleđivanja, kao odabran pravac prilikom implementacije ovde takođe ima uticaj na izmerenu vrednost metrike. Iako je vrednost blizu prihvatljivih normi,

uzimajući u obzir strukturu programskog koda, ovaj rezultat je i više nego prihvatljiv, jer i pored malog stepena nasleđivanja, skoro 50% metoda u sistemu koje se koriste su u suštini nasleđene.

U skladu sa ovim je i vrednost metrike AIF. U literaturi, manja vrednost ove metrike ujedno znači i da je sistem koji se testira manje kompleksan i lakši za održavanje. S obzirom na plitku strukturu nasleđivanja, ova vrednost je 0%, i kao takva neuporediva sa sistemima koji nisu u svojoj prirodi programska sprega i koji imaju duboko stablo nasleđivanja.

POF metrika, kao mera stepena polimorfizma, nije primenljiva na sisteme programske sprege i kao takva nije definisana za ovaj sistem.

6.1.2 Ocena kvaliteta rešenja na nivou nasleđivanja

Nasleđivanje je jedan od osnovnih principa OO dizajna. Pored prednosti kao što su ponovno iskorišćenje klase, koje se odnosi na dodavanje dodatnih karakteristika određenoj klasi, bez izmena same klase, nasleđivanje, ukoliko se intenzivno koristi može dovesti do pojave da programski kod postane težak za razumevanje i održavanje. U nastavku će biti dati rezultati DIT i SIX metrika za predloženi sistem.

Slika 6.2 prikazuje izmerene rezultate DIT metrike. U literaturi [Lorenz] je predložen prag za DIT metriku u vrednosti 6 za individualne klase, što usled plitkog stabla nasleđivanja, sve klase predložene programske podrške ispunjavaju.

Slika 6.2 Rezultati DIT metrike za predloženo programsko rešenje u odnosu na maksimalni prag predložen u literaturi

Sa stanovišta sistema programske podrške, velike vrednosti za DIT predstavljaju signal da se u sistemu nalaze kompleksni objekti, potencijalno teški za testiranje. Sa druge strane, male vrednosti DIT metrike upućuju na zaključak da sistem sadrži programski kod koji ne iskorišćava na pravi način prednost mehanizma nasleđivanja u OO dizajnu. S obzirom na to da analizirana i predložena programska sprega ima za glavni cilj laku proširivost i ograničenje realnog vremena, plitko stablo nasleđivanja je izabrano kao model za implementaciju. Stoga i izmerene vrednosti DIT metrike ukazuju na programsku podršku koja nema kompleksne objekte, teške za održavanje i testiranje.

Sa ovim u skladu, SIX metrika, čije rezultate prikazuje Slika 6.3, takođe pokazuje da sve klase u predloženom programskom rešenju ispunjavaju uslove predložene u literaturi [Lorenz] (prag od 15%). Prilikom merenja SIX metrikom, klase koje implementiraju sprežni sloj (engl. interface), tipične prilikom programiranja programskih sprega, nisu uzete u obzir jer ove klase usled mehanizma sprege, moraju da implementiraju sve metode osnovne klase, sto im automatski daje najlošiji rezultat. Ovo je mana same metrike, i stoga se u rezultatima SIX metrike nalazi manji broj klasa u odnosu na ostale metrike.

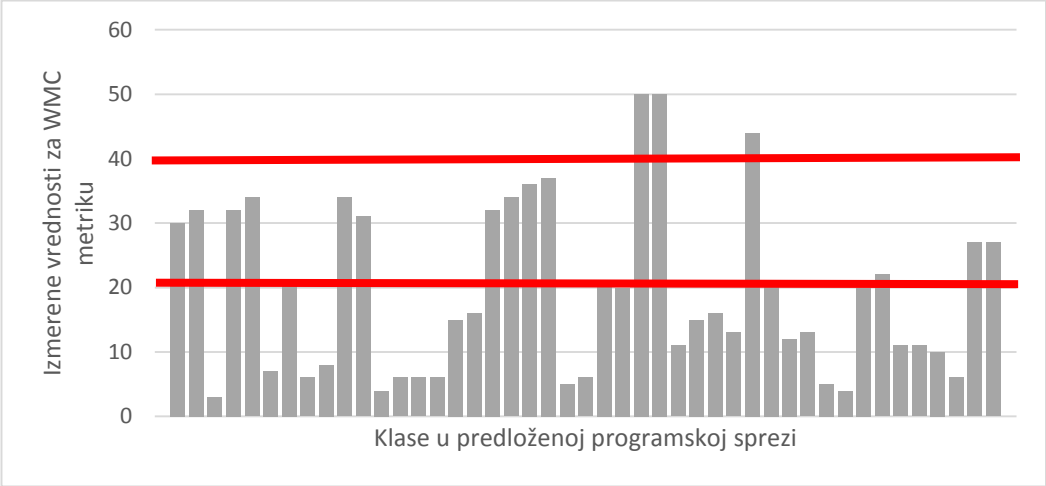
Slika 6.3 Rezultati SIX metrike za predloženo programsko rešenje u odnosu na maksimalni prag predložen u literaturi

6.1.3 Ocena kvaliteta rešenja na nivou klasa

Ova grupa metrika identifikuje karakteristike u okviru jedne klase, naglašavajući različite aspekte apstrakcije. Predložena programska podrška je analizirana sa sledeće tri metrike na nivou klasa:

- WMC – Ponderisane metode za klasu
- LCOM* (unapređena LCOM metrika) – Nedostatak kohezije
- RFC – Odziv klase

WMC meri kompleksnost klase i može se posmatrati kao indikator napora potrebnog za razvoj i održavanje klase. Klase koje imaju veliki broj metoda imaju uticaj na klase koje ih nasleđuju, jer od njih nasleđuju sve metode. U [Lorenz], predložen je prag od 20 do 40 metoda po klasi, i kao što prikazuje Slika 6.4, samo 3 klase iz celog predloženog rešenja ne ispunjavaju ovaj kriterijum (od ukupno 46 klasa). Ova činjenica govori u prilog tome da je predložena programska podrška pisana tako da se kompleksnost klasa držala u granicama prihvatljivog sa stanovišta kvaliteta objektno orijentisane programske sprege.



Slika 6.4 Rezultati WMC metrike za predloženo programsko rešenje u odnosu na maksimalne pragove predložene u literaturi

Namena LCOM* metrike je da izmeri koheziju klase, mereći broj zajedničkih atributa koji se koriste u različitim metodama klase. Drugim rečima, ova metrika meri kvalitet apstrakcije koju klasa predstavlja. Visoka vrednost LCOM* nagoveštava nedostatak kohezije, odnosno malu sličnost u okviru klase, tj. da se klasa sastoji od

nepovezanih objekata i da se kao takva može podeliti na više manjih klasa, koje imaju veći stepen kohezije. Slika 6.5 prikazuje izmerene vrednosti LCOM* metrike na sistemu koji se analizira. Izuzev pet klasa, najveći deo predložene programske podrške ima gotovo savršen odnos kohezije u svojim klasama.

Uzimajući u obzir da je predložena programska podrška, zapravo programska sprega, koja mora posedovati klase koje služe za spregu sa ostatkom sistema i koje sadrže isključivo metode za dobavljanje i postavljanje atributa klase (engl. getters and setters), ovakav rezultat je očekivan.

Slika 6.5 Rezultati LCOM* metrike za predloženo programsko rešenje

RFC je mera kompleksnosti klase koja se meri kroz broj metoda i količinu komunikacije između klasa. RFC se može smatrati i kao indikator potrebne količine vremena za testiranje i ispravljanje grešaka jer, što je veća RFC vrednost, to je veća kompleksnost klase zbog velike količine metoda koje se mogu prozvati. U literaturi ne postoji predlog apsolutnih pragova, koje programska podrška koja se testira mora da ispuni da bi posedovala traženi kvalitet. Iz tog razloga, sadržaj koji prikazuje Slika 6.6 se može posmatrati kao komparativni pregled kompleksnosti klasa u okviru predložene programske podrške.

Slika 6.6 Rezultati RFC metrike za predloženo programsko rešenje

6.1.4 Ocena kvaliteta rešenja na nivou metoda klasa

Atributi i metode klasa predstavljaju najfinije detalje jedne klase. Metrike koje ih analiziraju stoga i jesu važne prilikom ocene kvaliteta programske podrške, ali usled kompleksnih OO mehanizama, kao što su polimorfizam ili nasleđivanje, ovakve metrike se ne smeju posmatrati nezavisno od drugih metrika (na nivou klasa, ili sistema u celini). Najpoznatija metrika ovog tipa je broj linija programskog koda – LOC. U [Lorenz] je predložen prag LOC = 8, za objektno orijentisane programske jezike slične Java programskom jeziku koji je korišćen prilikom implementacije predložene programske podrške. Tabela 6.7 prikazuje opšte karakteristike predložene programske podrške uključujući i rezultate LOC metrike.

Karakteristika	Vrednost
Broj paketa	22
Broj klasa	46
Broj metoda	870
LOC (ukupno)	4542
LOC (po klasi)	98.73
LOC (po metodi)	5.22

Tabela 6.3 Karakteristike predložene programske podrške

6.2 Merenje performansi rešenja

Glavni cilj programskog rešenja opisanog u ovoj disertaciji je univerzalnost, kompletnost i platformska nezavisnost. Ove tri karakteristike, same po sebi nisu dovoljne ukoliko rešenje ne poseduje odgovarajuće performanse, koje se ogledaju u zadovoljavanju korisničkih očekivanja od programske podrške modernog TV prijemnika. Ova očekivanja se najčešće odnose na standardne TV funkcionalnosti, kao što su pronalaženje TV servisa, prelaz sa jednog na drugi TV servis ili vreme pokretanja samog TV uređaja, kao što je opisano u [Kooij] i [DSL Forum]. Uzimajući u obzir da moderni TV prijemnici, a posebno TV prijemnici bazirani na Android operativnom sistemu, osim standardnih digitalnih TV servisa, nude i dodatne usluge kroz IPTV servise i usluge video na zahtev (VOD), potrebno je potvrditi da se i ovi aspekti programske podrške izvršavaju na zadovoljavajućem nivou, kao što je definisano u [ATIS-IIF] i [ITU-T]. Tabela 6.4 prikazuje izmerene rezultate za tipične karakteristike opisane u [ATIS-IIF] i [ITU-T] koje se odnose na interakciju sa korisnikom, na jednoj eksperimentalnoj platformi opisanoj u poglavlju "Eksperimentalna potvrda predloženog programskog rešenja".

Karakteristika	**Očekivano vreme**	**Izmereno vreme**
Pronalaženje TV servisa (auto)	120 sec	110 sec[1]
Promena kanala (DVB > DVB)	2 sec	1.8 sec
Promena kanala (DVB > IP)	2 sec	1.6 sec
Promena kanala (IP > IP)	2 sec	1.7 sec
Pokretanje uređaja	10 sec	23 sec
Prikaz VOD sadržaja	2 sec	1 sec
Prikaz EPG sadržaja	2 sec	1 sec
Pretraga EPG sadržaja	/	1.5 sec
Pretraga VOD sadržaja	/	1.5 sec
NAPOMENE:	[1] Za 7 različitih frekvencija i 34 digitalna TV servisa	

Tabela 6.4 Izmerene performanse sistema na eksperimentalnoj platformi [Armada 1500 Pro]

Kao što se može videti u tabeli, sva vremena su u granicama očekivanih vrednosti, izuzev vremena pokretanja uređaja. Ovo vreme nije uzrokovano predloženim programskim rešenjem, već predstavlja ograničenje Android operativnog sistema, i svi uređaji bazirani na njemu se suočavaju sa sporim vremenom pokretanja uređaja.

Uzimajući u obzir da prilagođenje predloženog programskog rešenja ciljnoj fizičkoj platformi zahteva određeno vreme i napor, za testiranje performansi izabrana je [Armada 1500 Pro] platforma. Ova platforma predstavlja tipičan primer moderne, višejezgarne TV/STB platforme, a rezultati dobijeni na njoj se mogu smatrati kao reprezentativni za celu klasu ovakvih rešenja.

6.3 Testiranje rešenja od strane eksperata u oblasti potrošačke elektronike

Predloženo programsko rešenje, odnosno ceo Android4TV koncept je prikazivan u nekoliko navrata na dva najveća svetska sajma potrošačke elektronike: CES i IBC. CES je najveći svetski sajam na kojem se izlažu najsavremenija rešenja potrošačke elektronike iz oblasti digitalnih IT, telekomunikacionih i multimedijalnih sistema. IBC je najveći sajam u Evropi na kojem se izlažu najnoviji sistemi i uređaji iz oblasti multimedija i telekomunikacija. Kroz prikazane makete na ova dva sajma, tokom nekoliko godina, može se pratiti pravac razvoja predloženog rešenja i njegova evolucija tokom vremena.

Prva maketa predloženog programskog rešenja opisanog u ovoj disertaciji je prikazana na CES sajmu 2012. godine u Las Vegasu u saradnji sa kompanijom iWedia. Izložena maketa je imala sledeće karakteristike:

- Verzija Android operativnog sistema: 2.2 (Froyo).
- TV karakteristike:
    - Pronalaženje TV servisa
    - Manipulacija listama TV servisa
    - EPG funkcionalnost (trenutni/naredni EPG događaji i raspored EPG kanala)

    - Prikazivanje dodatnih sadržaja preko živog video toka u formi kartica (engl. widgets): vesti, internet pretraživač, vremenska prognoza
- Platforma: Bazirana na integrisanom kolu [Armada 1500] kompanije Marvell.
- Tip tjunera: DVB-T putem USB sprege.

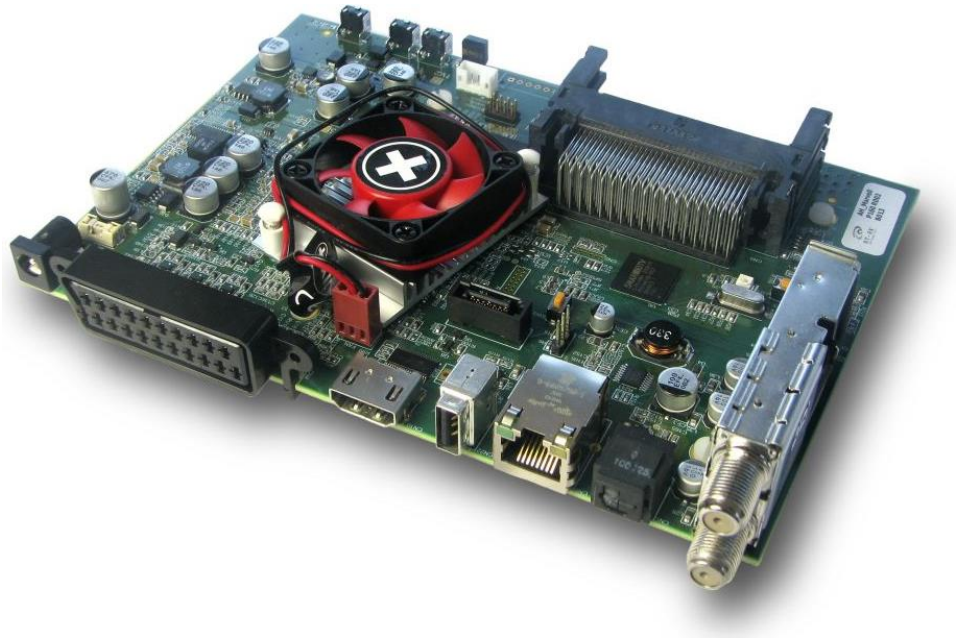
Posetioci su jednoznačno ocenili pokazan sistem kao veoma napredan, i koji na pogodan način integriše TV funkcionalnost u Android operativni sistem. Prikazane mogućnosti sistema u pogledu broja podržanih servisa digitalne televizije i njihovog kvaliteta su ocenjene pozitivno. Takođe, u okviru sajma se nalazi i internacionalna naučna konferencija, na kojoj je te godine rad [Vidakovic], koji opisuje osnovni koncept Android4TV i predstavlja osnovu ove disertacije, nagrađen posebnom Merit nagradom i izdvojen kao rad sa posebnim naučnim doprinosom u svom polju. Ipak, te godine pokazalo se da moderni TV prijemnici na tržištu potrošačke elektronike moraju imati podržane i dodatne napredne servise digitalne televizije, kao što su PVR, odnosno HbbTV.

Sledeća maketa predloženog programskog rešenja, koja predstavlja naredni evolucioni korak u razvoju Android4TV koncepta, prikazana je na IBC sajmu 2012. godine u Amsterdamu u saradnji sa kompanijom iWedia. Izložena maketa je imala sledeće karakteristike:

- Verzija Android operativnog sistema: 2.3 (Gingerbread).
- TV karakteristike (dodatno u odnosu na prethodnu maketu):
    - Instalacija TV servisa
    - PVR
    - Usluga odloženog gledanja (engl. time-shift)
    - HbbTV
    - Multimedijalna funkcionalnost
- Platforma: Bazirana na integrisanom kolu [Armada 1500] kompanije Marvell, koju prikazuje Slika 6.7.
- Tip tjunera: Integrisan DVB-S u okviru platforme.

Pored unapređenja verzije Android operativnog sistema, ova maketa donosi nekoliko značajnih funkcionalnih unapređenja u vidu naprednih TV aplikacija kao što

su PVR aplikacija, odnosno mogućnost odloženog gledanja TV servisa, ali i podrška za HbbTV stranice. Reakcije eksperata koji su prisustvovali prezentaciji se ogledaju u pozitivnoj oceni ideje, kao i postignute funkcionalnosti u vidu broja podržanih servisa digitalne televizije. Pokazan sistem, iako značajno unapređen u odnosu na prethodni sajam i dalje u sebi nije sadržao neke od standardnih servisa digitalne televizije, kao što su podrška za roditeljsku kontrolu, odnosno standarde tipa MHEG i kontrolisan pristup video sadržaju (CA).



Slika 6.7 Izgled [Armada 1500] bazirane platforme

Sledeća maketa predloženog programskog rešenja, koja predstavlja naredni evolucioni korak i naredni nivo unapređenja u razvoju Android4TV koncepta, ponovo je prikazana na CES sajmu, 2013. godine. Izložena maketa je imala sledeće karakteristike (poboljšanja u odnosu na prethodne makete):

- Verzija Android operativnog sistema: 4.0 (Ice Cream Sandwich).
- TV karakteristike (dodatno u odnosu na prethodnu maketu):
  - CI+
  - MHEG
  - TV aplikacija postaje osnovna aplikacija (engl. launcher)

- Platforma: Bazirana na integrisanom kolu [Armada 1500] kompanije Marvell.
- Tip tjunera: Integrisan DVB-T u okviru platforme.

Nova maketa, osim unapređenja Android operativnog sistema uvodi nova unapređenja u vidu sistemske podrške za MHEG standard i kontrolisan pristup, ali i postavku TV aplikacije, kao podrazumevane osnovne aplikacije (Slika 6.8 prikazuje izgled pokazane makete). Eksperti su pored pohvala na kvalitet pokazanog rešenja, ukazali na činjenicu da predložena programska podrška a i sam koncept, ukoliko žele da budu konkurentni na modernom tržištu TV/STB prijemnika, treba da podrže i funkcionalnosti kao što su roditeljska kontrola pristupa, odnosno DLNA podrška.

Slika 6.8 Izgled makete pokazane na sajmu

Naredne makete predloženog programskog rešenja prikazane su na IBC sajmu 2013, odnosno CES sajmu početkom 2014. godine. Ova rešenja, osim unapređenja verzije Android operativnog sistema na poslednju aktuelnu verziju, dodaju neophodne funkcionalnosti koje su nedostajale u prethodnim rešenjima, kao što su:

- Roditeljska kontrola
- DLNA

- Podrška za udaljene upravljačke aplikacije

Osim dodatne funkcionalnosti, programsko rešenje je demonstrirano na maketama koje su bazirane na Marvell i Broadcom platformama, čime je potvrđen princip platformske nezavisnosti rešenja. Reakcije eksperata koji su evaluirali pokazane makete je veoma dobar, sa stanovišta količine podržanih servise digitalne televizije, ali i kvaliteta njihove integracije u Android operativni sistem, što je veoma bitan faktor za proizvod namenjen širokom krugu korisnika. Slika 6.9 prikazuje izgled korisničke sprege TV aplikacije pokazane na IBC/CES sajmovima 2013. i 2014. godine.

Slika 6.9 Izgled korisničke sprege TV aplikacije pokazane na maketama na CES/IBC sajmovima

6.4 Poređenje sa drugim rešenjima

U cilju poređenja sa postojećim rešenjima, analizirano je nekoliko postojećih proizvoda. Na tržištu TV/STB prijemnika baziranih na Android operativnom sistemu postoji znatno više proizvoda, ali detalji i karakteristike mnogih nisu dostupni. Osim toga, značajan broj proizvoda deli izuzetno slične performanse i karakteristike. Odabrani su sledeći proizvodi sa poznatim karakteristikama kao tipični predstavnici komercijalnih TV/STB prijemnika baziranih na Android operativnom sistemu:

- Hibridni Android bazirani STB uređaj *SRTAN4*, kompanije Strong, [ProdAndrStrong]
- GoogleTV STB posredinik *Logitech Revue*, kompanije Logitech [ProdGTVLogitech]

Uređaj *SRTAN4* kompanije Strong predstavlja hibridno rešenje Android STB prijemnika. Ovi uređaji, osim IPTV dela, poseduju komponentu fizičke arhitekture tjuner - koja predstavlja primarni izvor DVB transportnog toka. Ovaj uređaj poseduje podržanu većinu standardnih digitalnih TV servisa, izuzev uslovnog pristupa video sadržaju i roditeljske kontrole. Mana ovog uređaja, osim što implementira samo podskup standardnih TV servisa, je to što je TV aplikacija samo jedna od aplikacija instaliranih na sistemu, i ravnopravna u odnosu na sve ostale. Stoga napredne funkcije Android operativnog sistema, kao što su pretraga podataka ne konsultuju podatke koji dolaze iz DTV sveta.

Uređaj *Logitech Revue* kompanije Logitech, predstavlja tipičan primer STB uređaja baziranog na GoogleTV platformi, koji implementira posrednički mehanizam. Upotreba ovog mehanizma uslovljava da ovaj uređaj ne poseduje fizičku komponentu za manipulaciju DVB transportnim tokom – tjuner. Ovaj uređaj TV servise dobija kroz ulaznu HDMI video spregu (kao što pokazuje Slika 3.3). Ovo za posledicu ima da mnogi digitalni TV servisi nisu podržani od strane ovog uređaja, dok performanse, kao što su vreme promene kanala, ili pretraga frekventnog opsega zavise od STB uređaja sa kojim je *Logitech Revue* povezan. I pored ovih mana, kao i svaki GoogleTV baziran uređaj, *Logitech Revue* poseduje visok nivo integracije TV podataka u Android operativni sistem, koji se ogleda u sistemskoj podršci za pretragu.

U daljem tekstu će se odabrana rešenja za analizu porediti po sledećim kriterijumima:

- mogućnostima/podržanim funkcijama,
- performansama i
- kvalitetu programske sprege.

U cilju preglednosti, rezultati se prikazuju tabelarno. Tabela 6.5 prikazuje mogućnosti razmatranih rešenja. Pod mogućnostima sistema u ovom kontekstu se razmatra broj podržanih standardnih digitalnih TV servisa (prethodno definisanih u poglavlju “Mera kompletnosti programske sprege za servise digitalne televizije”).

Uz ove servise, dodata je i kolona za pretragu DTV sadržaja, kao indikator stepena integracije podataka iz DVB transportnog toka u Android operativni sistem.

Rešenje / Funkcija	Logitech Revue	SRTAN4	Android4TV
SCAN	✓[1]	✓	✓
EPG	✓[2]	✓	✓
SUB	✓[3]	✓	✓
TTX	✓[3]	✓	✓
REM	×	×	✓
PVR	✓	✓	✓
CA	×	×	✓
PARENTAL	×	×	✓
VOD	✓	✓	✓
INPUT	✓[4]	×	✓
Pretraga EPG sadržaja	✓[2]	×	✓
NAPOMENE:	[1] Preko predefinisanog ZIP koda [2] Samo putem interneta (ne iz DVB transportnog toka) [3] Ukoliko STB posrednik poseduje podršku [4] Samo HDMI tip ulazne sprege. Drugi tipovi ulaznih sprega nisu podržani.		

Tabela 6.5 Pregled mogućnosti pojedinih rešenja

Iz tabele se vidi da rešenje Android4TV pokriva širok spektar mogućnosti, i da ga je moguće primeniti u velikom broju scenarija. GoogleTV rešenja, u koja spada i *Logitech Revue*, pokrivaju takođe većinu TV servisa, ali se putem posredničkog mehanizma oslanjaju na mogućnosti STB uređaja sa kojim su povezani. Ovo znači da ne postoji sistemska programska podrška u ovakvim uređajima i kvalitet korisničkog ugođaja zavisi od STB prijemnika koji je izvor informacija. Ovo predstavlja ozbiljnu manu ovakvih rešenja. STB uređaji bazirani na Android OS, kao što je *SRTAN4*

najčešće implementiraju samo podskup funkcionalnosti i poseduju nizak stepen integracije podataka iz TV sveta u Android operativni sistem.

Performanse uređaja se procenjuju na osnovu nekoliko pokazatelja, prethodno opisanih u poglavlju "Merenje performansi rešenja"). Ovi pokazatelji se odnose na standardne TV funkcionalnosti, kao što su pronalaženje TV servisa, prelazak sa jednog na drugi TV servis ili vreme pokretanja samog TV uređaja, kao što je opisano u [Kooij] i [DSL Forum]. Pregled performansi pojedinih rešenja prikazuje Tabela 6.6.

Rešenje / Karakteristika	Logitech Revue	SRTAN4	Android4TV
Pronalaženje TV servisa (auto)	/[1]	157 sec	110 sec
Promena kanala (DVB > DVB)	/[2]	2.5 sec	1.8 sec
Promena kanala (DVB > IP)	/[2]	/	1.6 sec
Promena kanala (IP > IP)	/[2]	/	1.7 sec
Pokretanje uređaja	41 sec	39 sec	23 sec
Prikaz VOD sadržaja	2.1 sec	3.4 sec	1 sec
Prikaz EPG sadržaja	1.5 sec[3]	1 sec	1 sec
Pretraga EPG sadržaja	1.5 sec[3]	/	1.5 sec
Pretraga VOD sadržaja	2 sec	1.2 sec	1.5 sec
NAPOMENE:	[1] Predefinisana lista kanala određena ZIP kodom [2] Zavisi isključivo od mogućnosti i performansi STB posrednika [3] Samo putem interneta (ne iz DVB transportnog toka)		

Tabela 6.6 Pregled performansi pojedinih rešenja

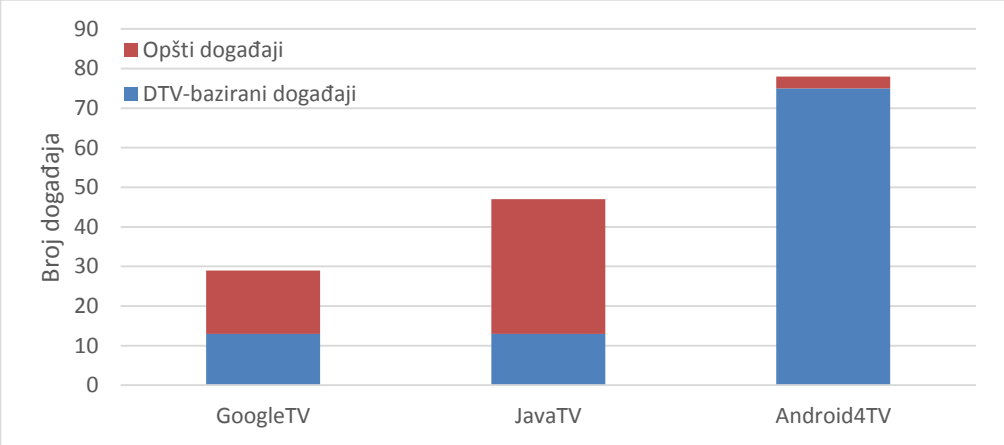
Iz tabele se vidi da je rešenje Android4TV konkurentno sa drugim rešenjima na tržištu Android baziranih STB/TV uređaja. Treba naglasiti da sva tri upoređena rešenja ne koriste iste platforme, i stoga ovo poređenje nije na nivou programske podrške uređaja već celokupnog sistema. Takođe, *Logitech Revue* rešenje, zbog posredničkog mehanizma, svoje performanse nasleđuje od izvornog STB uređaja.

Za potrebe poređenja kvaliteta programske sprege Android4TV rešenja, isto je potrebno uporediti sa drugim standardnim Java baziranim rešenjima. Kao dva tipična primera Java programskih sprega za DTV funkcionalnost, razmatrane su dve standardizovane programske sprege (API): GoogleTV i JavaTV. Kao parametar poređenja, može se koristiti mera kontrolabilnosti (opisana u poglavlju "Mera kontrolabilnosti programske sprege za servise digitalne televizije"). Ova mera se oslanja na činjenicu da kvalitet TV aplikacije koja se može razviti na Android OS, direktno zavisi od kvaliteta Java programske sprege koja implementira DTV funkcionalnost. Jedan od važnih faktora prilikom ocene kvaliteta ovakve sprege se ogleda u količini informacija, odnosno poruka koje se razmenjuju kroz slojeve programske podrške. Tabela 6.7 daje pregled broja događaja u programskim spregama sa kojima se vrši poređenje.

Rešenje / Događaji	GoogleTV	JavaTV	Android4TV
SCAN	2	1	10
EPG	1	4	7
SUB	2	/	1
TTX	/	/	2
REM	/	/	1
PVR	6	/	40
CA	/	4	2
PARENTAL	/	/	3
Promena kanala	4	4	7
INPUT	1	/	2
Opšti događaji	*13	+34	3
NAPOMENE:	* Događaji vezani za arhitekturu programskog rešenja + Događaji koji nisu usko vezani za DTV, ali se donekle preklapaju		

Tabela 6.7 Pregled broja događaja programske sprege pojedinih Java baziranih TV programskih sprega

Uzimajući u obzir količinu informacija koje obrađuje i broja stanja kroz koje prolazi sistemska TV programska podrška prilikom redovnog izvršavanja, da bi se implementirala TV aplikacija zadovoljavajućeg kvaliteta, potrebno je razmeniti srazmerno veliku količinu poruka. U cilju tačnijeg poređenja, a uzimajući u obzir da je domen poređenja DTV funkcionalnost, posebno su izdvojeni opšti događaji, od događaja koji su usko vezani za DTV funkcionalnost. Slika 6.10 daje grafički prikaz odnosa broja opštih i DTV-baziranih događaja u programskim spregama koje se porede. Rešenje koje ima bolji odnos u korist DTV-baziranih događaja se smatra kontrolabilnijim i prilagođenijim upotrebi u modernom TV/STB prijemniku.

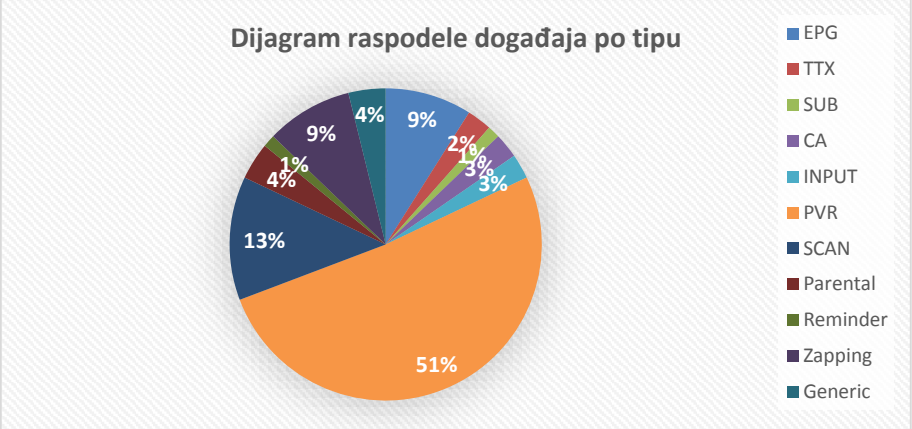


Slika 6.10 Grafički prikaz odnosa broja opštih i DTV-baziranih događaja pojedinih Java baziranih TV programskih sprega

Iz prethodne tabele i slike, može se zaključiti da Android4TV rešenje sadrži veći broj i bolji odnos DTV-baziranih rešenja u odnosu na komercijalno raspoloživa rešenja. Slika 6.11 prikazuje dijagram raspodele broja događaja po tipu u okviru Android4TV rešenja. Očekivano, PVR događaji dominiraju u ukupnom broju događaja, usled kompleksnosti modernih PVR aplikacija i broja različitih stanja u kojem takva aplikacija može da se nađe.

Na osnovu gornje analize, može se zaključiti da je Android4TV najkompletnije rešenje u smislu broja podržanih servisa digitalne televizije, koje je moguće primeniti na širokom spektru različitih platformi i koje nije vezano za posebnu verziju Android operativnog sistema. Osim platformske nezavisnosti i kompletnosti, Android4TV

pruža i integraciju DTV baziranih podataka u jezgro Android operativnog sistema kroz podršku za sistemsku pretragu DVB podataka.



Slika 6.11 Dijagram raspodele događaja spram tipa

Sa stanovišta performansi, Android4TV je u rangu sa najboljim postojećim rešenjima, dok sa stanovišta kvaliteta programske sprege ovo rešenje pokazuje najveći nivo prilagođenosti funkcionalnostima DVB servisa uz visok nivo kontrolabilnosti i razmene informacija kroz postojeće slojeve predložene programske arhitekture.

POGLAVLJE 7.

ZAKLJUČAK

Usled sve prisutnije digitalizacije svih oblasti delovanja, polje primenjene elektronike je danas obeleženo ponudom jedinstvenih digitalnih servisa koji su krajnjem korisniku raspoloživi na različitim uređajima potrošačke elektronike, a posebno kroz moderne televizijske prijemnike. Savremena tehnička rešenja TV prijemnika koriste Android operativni sistem kao osnovu za obezbeđivanje savremenih digitalnih TV servisa. U ovoj disertaciji fokus je na istraživanju mogućih pristupa za proširenje Android operativnog sistema servisima digitalne televizije, kao i predlog rešenja koje postiže rad u realnom vremenu i zadovoljava zadate zahteve u pogledu složenosti, kompletnosti proširenja ali i platformske nezavisnosti i univerzalnosti.

U okviru istraživanja analizirana su postojeća rešenja sa ciljem da se ustanovi trenutno stanje na tržištu. Rezultat pretrage pokazuje da trenutno ne postoji komercijalno raspoloživo rešenje koje obezbeđuje integraciju svih standardnih digitalnih TV servisa i njihovu spregu sa ostatkom Android OS, koje se ogleda u korišćenju naprednih funkcija ovog operativnog sistema u kombinaciji sa informacijama koje dolaze iz TV sveta. Tokom analize identifikovani su okviri rešenja sa stanovišta očekivane kompleksnosti platforme i skupa digitalnih servisa od interesa i odabran je skup ciljnih platformi ([Armada 1500 Pro], [BCM7425], [MPQ8064] i [STiH 416] sa karakteristikama koje prikazuje Tabela 5.4) za eksperimentalnu potvrdu platformske nezavisnosti predloženog rešenja.

Pretraga baze patenata pokazala je da je oblast integracije digitalnih TV servisa u ugrađenim sistemima i platformama kao što je Android veoma aktuelna, i da mnoge vodeće kompanije iz oblasti telekomunikacija, IT industrije i proizvodnje digitalnih TV prijemnika, ulažu značajne napore u razvoj takvih sistema. Prikupljeno znanje je iskorišćeno tokom zaštite inovacije istraživanja odgovarajućim patentnim prijavama.

Postavljene su i mere za ocenu kvaliteta i performansi realizovanog rešenja, i realizovani su mehanizmi za merenje istih. Odabrane mere omogućuju ocenu kvaliteta predložene programske podrške sa stanovišta objektno orijentisane paradigme, brzine odziva i kompletnosti. Inovacije u okviru istraživanja su zaštićene sa dve nacionalne patentne prijave. Rešenje je imenovano zaštićenim imenom: *Android4TV*.

Razvoj je rezultovao programskom podrškom koja omogućuje standardne digitalne TV servise u Android operativnom sistemu. Osim aplikativne programske sprege pisane u Java programskom jeziku, koja omogućuje razvoj TV aplikacija, istraživanje je rezultovalo predlogom sistemskih izmena jezgra Android operativnog sistema neophodnih za napredne TV funkcije, ali i unapređenjem bazičnih mehanizama, kao što je IPC, u domenu prenošenja klasne hijerarhije.

Rezultati merenja odabranih indikatora kvaliteta su poređena sa postojećim rešenjima. Analiza je pokazala da je Android4TV integralno, univerzalno i kompletno rešenje koje je u stanju da obezbedi sve standardne TV servise, uz odgovarajući nivo kontrolabilnosti od strane glavne TV aplikacije i obezbedi spregu podataka DVB transportnog toka sa jezgrom Android operativnog sistema, uz obezbeđivanje odgovarajućeg nivoa kvaliteta programske sprege.

Kao dokaz disertacije realizovana je maketa zasnovana na [Armada 1500 Pro] platformi. U skladu sa očekivanjima, sistem radi u realnom vremenu, te je ograničenje u pogledu složenosti zadovoljeno. Osim [Armada 1500 Pro] platforme, predloženo programsko rešenje je realizovano i na dodatnim maketama baziranim na drugim platformama kao što su [BCM7425], [MPQ8064] i [STiH 416], koje osim platformskih razlika sadrže i razlike u arhitekturi korišćenih integrisanih kola ali i prisutnih DVB sprega, čime je potvrđena platformska nezavisnost i univerzalnost rešenja. Makete su u prethodnih nekoliko godina, počev od 2012, prikazane na nekoliko najvećih svetskih sajmova potrošačke elektronike, kao što su IBC u Amsterdamu i CES u Las Vegasu, gde je odziv eksperata u pogledu ideje, kvaliteta i performansi bio veoma pozitivan.

Naredni mogući korak istraživanja, zasnovanog na rezultatima ove disertacije je integracija i proširenje postojeće programske podrške sa interaktivnim TV standardima kao što je HbbTV. Osim omogućavanja prikaza HbbTV aplikacija, potrebno je istražiti mehanizme i potrebne izmene u jezgru Android operativnog sistema, kao što je Internet pretraživač, da bi se jedan ovako kompleksan interaktivni TV servis na pravi način podržao. Osim integracije, potrebno je i istražiti kako spregnuti podatke koji se nalaze u okviru HbbTV stranica sa ostatkom Android ekosistema i načiniti ih pretraživim. Ovo je poseban izazov uzimajući u obzir da su HbbTV stranice bazirane na CE-HTML standardu i najčešće pisane dinamičkim programskim jezicima, kao što je JavaScript.

Moguća su i dalja unapređenja u pogledu proširenja programskog rešenja sa funkcionalnošću dodatnog ekrana (engl. second-screen). Funkcionalnost dodatnog ekrana se odnosi na upotrebu drugih, najčešće mobilnih, uređaja kao što su pametni telefoni i tableti. Ovi uređaji pružaju interaktivne funkcije tokom linearnog sadržaja koje se u isto vreme prikazuju na centralnom TV prijemniku.

POGLAVLJE 8.

LITERATURA

[AADSP1] V. Kovačević, M. Popović, M. Temerinac, N. Teslić, "*Arhitekture i algoritmi digitalnih signal procesora I*", FTN, Novi Sad, 2005.

[Abe] K. Abe and N. Sugita, "*Distances between strings of symbols-Review and remarks,*" Proc. ICPR, pp. 172-174, 1982.

[Ableson] F. Ableson, C. Collins, R. Sen, "*Unlocking Android,*" Manning Publications Co., Greenwich, CT, USA, 2009.

[Abreu1] F. Brito e Abreu, W. Melo, "*Evaluating the impact of Object-Oriented Design on Software Quality,*" Proceedings of 3rd International Software Metrics Symp., Berlin, 1996

[Abreu2] F. Abreu, R. Carapuça, "*Object-Oriented Software Engineering: Measuring and Controlling the Development Process,*" Proceedings of the 4th International Conference on Software Quality, McLean, VA, USA, 1994.

[Abreu3] F. Abreu, M. Goulão, R. Esteves, "*Toward the Design Quality Evaluation of Object-Oriented Software Systems,*" Proceedings of the 5th International Conference on Software Quality, Austin, Texas, USA, 1995.

[Abreu4] F. Abreu, S. Esteves, M. Goulao, "*The Design of Eiffel Programs: Quantitative Evaluation Using the MOOD Metrics,*" Proceedings of TOOLS'96, Santa Barbara, CA, USA, 1996.

[Al-Ja'afer1] J. Al-Ja'afer, K. Sabri, "*Metrics for Object Oriented Design (MOOD) to Assess Java Programs,*" Internation Arab Conference on Information Technology (ACIT'04), Algeria, 2004.

[Al-Ja'afer2] J. Al-Ja'afer, K. Sabri, "*Chidamber-Kemerer (CK) and Lorenze-Kidd (LK) Metrics to Assess Java Programs,*" International Workshop on Software System (IWSS'04), Turkey, 2004.

[Android SDK] Android SDK, http://developer.android.com/sdk/index.html

[Archer] C. Archer, M. Stinson, "*Object-Oriented Software Measures,*" Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report CMU/SEI-95-TR-002, 1995.

[Armada 1500] http://www.marvell.com/digital-entertainment/armada-1500/

[Armada 1500 Pro] http://www.marvell.com/digital-entertainment/armada-1500-pro/

[ATIS-IIF] ATIS-IIF, ATIS-0800004, "A Framework for QoS Metrics and Measurements supporting IPTV Services ", October 2006.

[Baeza-yates] R. Baeza-yates, G. Navarro, "*Fast Approximate String Matching in a Dictionary,*" Proc. SPIRE'98. IEEE CS Press. pp. 14–22, 1998.

[BCM7425] http://www.broadcom.com/products/Cable/Cable-Set-Top-Box-Solutions/BCM7425

[Benoit] H. Benoit, "*Digital Television: Satellite, Cable, Terrestrial, IPTV, Mobile TV in the DVB Framework,*" 3rd ed. Burlington, MA: Focal Press, 2008.

[Brandao] R.R. de Mello Brandao, G.L. de Souza Filho, C.E.C.F. Batista, and L.F. Gomes Soares, "*Extended Features for the Ginga-NCL Environment: Introducing the LuaTV API,*" 19th International Conference on Computer Communications and Networks (ICCCN), Zurich, August 2010, pp. 1-6.

[Bunke] H. Bunke and A. Sanfeliu, eds., "S*yntactic and Structural Pattern Recognition: Theory and Applications,*" World Scientific Publishing Co., 1990.

[Cartwright] M. Cartwright, M. Shepperd, "*An Empirical Investigation of Object Oriented Software in Industry,*" Dept. of Computing, Talbot Campus, Bournemouth University Technical Report TR 96/01, 1996.

[Chidamber1] S. R. Chidamber, C. F. Kemerer, "*A metric suite for object oriented design,*" IEEE Transactions on Software Engineering, pp. 476–493, 1994.

[Chidamber2] S. R. Chidamber, D. P. Darcy, C.F. Kemerer, "*Managerial use for object oriented software metrics: an exploratory analysis*," IEEE Transactions on Software Engineering, 24(8), pp629-639, Aug 1998.

[Daly] J. Daly, A Brooks, J. Miller, M. Roper, M. Wood, "*Evaluating inheritance Depth on the Maintainability of Object-Oriented Software*," Empirical SE 1(2) Feb 1996.

[DSL Forum] DSL Forum TR-126, "Triple-play Services Quality of Experience (QoE) Requirements", 13 December 2006.

[EPO] *European Patent Office* – EPO, www.epo.org

[ETSI1] ETSI EN 300 472, Digital Video Broadcasting (DVB); Specification for conveying ITU-R System B Teletext in DVB bitstreams

[ETSI2] ETSI EN 300 468, Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems

[ETSI3] ETSI EN 300 707, Electronic Programme Guide (EPG); Protocol for a TV Guide using electronic data transmission

[ETSI4] ETSI EN 300 743, Digital Video Broadcasting (DVB); Subtitling systems

[ETSI5] ETSI TR 101 288, Television systems; Code of practice for an Electronic Programme Guide (EPG)

[ETSI6] ETSI TS 102 367, Digital Audio Broadcasting (DAB); Conditional access

[ETSI7] ETSI TS 102 816, Digital Video Broadcasting (DVB); Personal Video Recorder (PVR)/Personal Data Recorder, (PDR) Extension to the Multimedia Home Platform

[ETSI8] ETSI TS 101 211, Digital Video Broadcasting (DVB); Guidelines on implementation and usage of Service Information (SI)

[ETSI9] ETSI TR 101 290, Digital Video Broadcasting (DVB); Measurement guidelines for DVB systems

[Ferrate] A. Ferrate, A. Surya, D. Lee, M. Ohye, P. Carff, S. Shen, S. Hines, "*Building Web Apps for Google TV*,", O'Reilly Media, Inc., 2011.

[Gamma] E. Gamma, R. Helm, R. Johnson, J.M. Vlissides, "*Design Patterns: Elements of Reusable Object-Oriented Software*," Addison-Wesley Professional, 1994.

[Gargenta] M. Gargenta, "*Learning Android: Building Applications for the Android Market*,", O'Reilly Media, Inc., 2011.

[GoogleTV] GoogleTV, http://www.google.com/tv/

[Harrison1] R. Harrison, S. Counsell, R. Nithi, "*An Overview of Object-Oriented Design Metrics,*" Proc. of the conference on Software Technology and Engineering Practice (STEP), IEEE Press, pp. 230-237, Jul 1997.

[Harrison2] R. Harrison, S. Counsell, R Nithi, "*An evaluation of the MOOD Set of Object-Oriented Software Metrics,*" IEEE Transaction on Software Engineering, Vol. 24, No. 6,June 1998.

[Henderson] B. Henderson-Sellers, "*Object-Oriented Metrics Measures of Complexity,*" Upper Saddle River, NJ: Prentice Hall, 1996.

[Hendrawan] Hendrawan, N. S. Sugiharto, D. Hermawan, "*Design and Implementation of Interactive Mobile TV over Hybrid Network of DVB Broadcast Network and Unicast Network,*" 6th International Conference on Telecommunication Systems, Services, and Applications (TSSA), Dec. 2011, pp. 288-291.

[ITU-T] ITU-T FG IPTV-C-0411, "IPTV QoS/QoE Metrics"

[Java TV] Java TV, http://www.oracle.com/technetwork/java/javame/javatv/overview/getstarted/index.html

[Kooij] R. Kooij, K. Ahmed1, K. Brunnström "Perceived Quality of channel zapping," Proceedings of the Fifth LASTED International Conference COMUNICATION SYSTEMS AND NETWORKS, August 2006.

[Kuzmanovic] N. Kuzmanovic, T. Maruna, M. Savic, G. Miljkovic, D. Isailovic, "*Google's Android as an application environment for DTV decoder system,*" 14th IEEE International Symposium on Consumer Electronics (ISCE), June 2010, pp. 1-5.

[Kuzmanovic2] N. Kuzmanovic, V. Mihic, T. Maruna, M. Vidakovic, N. Teslic, "*Hybrid broadcast broadband TV implementation in java based applications on digital TV devices,*" Consumer Electronics, IEEE Transactions on, vol.58, no.3, pp.1056,1062, August 2012.

[Levenshtein] V. Levenshtein, "*Binary Codes Capable of Correcting Deletions, Insertions and Reversals,*" Soviet Physics-Doklady, vol. 10, no. 8, pp. 707-710, 1966.

[Lorenz] M. Lorenz, J. Kidd, "*Object-Oriented Software Metrics,*". Englewood, NJ: Prentice Hall, 1994.

[LPRS] V. Kovačević, "*Logičko projektovanje računarskih sistema*", FTN, Novi Sad, 1993.

[Lu] Y. S. Lu, C. H. Lee, H. Y. Weng, Y. M. Huang, "*Design and implementation of digital TV widget for Android on multi-core*

platform," International Computer Symposium (ICS), December 2010, pp. 576-580.

[Lucas] A. dos Santos Lucas, S.D. Zorzo, "*Personalization for Digital Television Using Recommendation System strategy,*" 16th International Conference on Systems, Signals and Image Processing (IWSSIP), Chalkida, June 2009, pp. 1-4.

[Lukac] Z. Lukac, M. Radonjic, B. Veris, T. Maruna, N. Kuzmanovic, "*The experience of implementing a Hybrid Broadcast Broadband Television on network enabled TV set,*" 34th International Convention MIPRO, May 2011, pp. 840-844.

[Lukic] N. Lukic, N. Teslic, T. Maruna, V. Mihic, "*A java API interface for the search of DTV services in embedded multimedia devices,*" Consumer Electronics, IEEE Transactions on, vol.59, no.4, pp.875,882, November 2013

[McCabe] T. J. McCabe, "*A Complexity Measure*," IEEE Transactions on Software Engineering, 2(4), 308-320, 1976.

[Miljkovic] G. Miljkovic, V. Mihic, M. Ristic, V. Kovacevic, "*DTV linux device abstraction for embeded systems,*" 14th IEEE International Symposium on Consumer Electronics (ISCE), June 2010, pp. 1-6.

[Moon] S. P. Moon, J. W. Kim, K. H. Bae, J. C. Lee, D. W. Seo, "*Embeded Linux implementation on a commercial digital TV system,*" IEEE Transactions on Consumer Electronics, vol. 49, no. 4, pp. 1402-1407, Nov. 2003.

[Morris] S. Morris, A. Smith-Chaigneau, "*Interactive TV Standards: A Guide to Mhp, Ocap, and JavaTV*", Focal Press, 2005.

[MPEG-2] ISO/IEC 13818-1:2000, International Standard, MPEG-2 system specification

[MPQ8064] http://www.qualcomm.com/media/blog/2013/06/17/snapdragon-mpq8064-processor-now-part-snapdragon-600-series

[Nielsen] The Nielsen Company, "*Consumer Electronics Ownership Blasts Off in 2013,*" September, 2013. http://shar.es/QwV8F

[Noergaard] T. Noergaard, "*Demystifying Embedded Systems Middleware,*" Amsterdam: Newnes, 2011.

[Papp] I. Papp, " *Prilog rešenju obrade govornog signala korišćenjem mikrofonskog niza*", Doktorska disertacija, 2009.

[PatBiap] Biap Systems Inc, "*Self-contained mini-applications system and method for digital television,*" WIPO patent, WO2007035514 A2, 2007.

[PatGoogle] Google Inc., "*Safe browser plugins using native code modules,*" USPTO patent, US20100122271 A1, 2010.

[PatGoogle2] Google Inc., "*Method and system for executing applications using native code modules*," USPTO patent, US20100017461 A1, 2010.

[PatHP] Hewlett-Packard Development Company, L.P., "*Method for mapping procedural C++ code to java object-oriented classes*," USPTO patent, US6886172 B2, 2005.

[PatIBM] International Business Machines Corporation, "*Batch dispatch of java native interface calls*," USPTO patent, US20120167067 A1, 2012.

[PatImerj] Imerj Llc, "*Cross-environment communication framework*," WIPO patent WO2012044558 A2, PCT/US2011/053130, 2012.

[PatLg] Lg Electronics Inc., "*Multimedia device having operating system capable of processing multiple graphic data and method for controlling the same*," EPO patent, EP2475184 A1, 2012.

[PatLg2] Lg Cns Co., Ltd., "*Smart set-top box and operating method for smart service and digital television service using single operating system*," WIPO patent, WO2012102568 A2, 2012.

[PatNine] Nine of Technology Co., Ltd. of Guangdong, "*Android system of set top box (STB)*," SIPO patent, CN103200448 A, 2013.

[PatPhilips] Koninkl Philips Electronics Nv, Philips Corp, "*Method for concurrently presenting multiple content types in a tv platform*," WIPO patent, WO2004047433 A1, 2004.

[PatSichuan] Sichuan Changhong Electric Co., Ltd., "*Interaction method of browser and hardware*," SIPO patent, CN103019679 A, 2013.

[PatSun] Sun Microsystems, Inc., "*Application programming interface for connecting a platform independent plug-in to a web browser*," USPTO patent, US7069562 B2, 2006.

[PatSun2] Sun Microsystems, Inc., "*Inter-process communication using different programming languages*," USPTO patent, US7444619 B2, 2008.

[PatTcl] Tcl Group Co., Ltd., "*Method for integrating Android application system in television system based on Android inner core*," SIPO patent, CN102541558 A, 2012.

[PatTianjin] Tianjin platinum record Kunishige Electronic Technology Development Co., Ltd. of Guangdong, "*Android application Loader download management system facing broadcast and TV service providers*," SIPO patent, CN202818526 U, 2013.

[PatYahoo] Yahoo! Inc., "*System for packaging native program extensions together with virtual machine applications*," USPTO patent, US 8,607,224 B2, 2010.

[PatYahoo2] Yahoo! Inc., "*System for packaging native program extensions together with virtual machine applications,*" USPTO patent, US 8,607,224 B2, 2010.

[ProdAndrFT] ForceTech, Android HD STB, http://www.forcetech.net/en/product/Android.html

[ProdAndrNevron] Nevron, FaSTBox, http://www.nevron.eu/products/iptv-accessories/iptv-set-top-box

[ProdAndrPeerTV] PeerTV, eTV SmartTV station, http://www.peertv.com/#!what-can-i-do-with-etv/cjg9

[ProdAndrStrong] Strong, SRTAN4, http://www.android4tv.com.au/

[ProdGTVAsus] Asus, *Asus Cube*, http://promos.asus.com/US/asuscube/

[ProdGTVHisense] Hisense, *Hisence Pulse*, http://www.hisense-usa.com/pulse/

[ProdGTVLG] LG, *LG SmartTV with GoogleTV*, http://www.lg.com/us/lggoogletv

[ProdGTVLogitech] Logitech, *Logitech Revue*, www.logitech.com/en-us/smartTV/revue

[ProdGTVNetGear] NetGear, *NeoTV Prime with Google TV*, http://www.netgear.com/landing/stream/tv/#googletv

[ProdGTVSFR] SFR, *SFR TV Decoder*, http://adsl.sfr.fr/television/decodeur-tv-sfr-avec-google-play/

[ProdGTVSony] Sony, *Sony Internet TV with Google TV*, http://www.sony.com/

[ProdGTVSony2] Sony, *Sony Internet TV Blu-ray Disc Player with Google TV*, http://www.sony.com/

[ProdGTVSony3] Sony, *Internet Streaming Player Powered by Google TV*, http://www.sony.com/

[ProdGTVVisio] Vizio, *Vizio Co-Star*, http://store.vizio.com/co-star

[SPP] M. Popović, "*Sistemska programska podrška*", FTN, Novi Sad, 2004.

[STiH 416] http://www.st.com/web/catalog/mmc/FM131/SC999/SS1633/PF253155

[Ukkonen] E. Ukkonen, "*Finding Approximate Patterns in Strings,*" J. Algorithms, vol. 6, pp. pp. 132-7,1985.

[USPTO] *US Patent and Trademark Office* – USPTO, www.uspto.gov

[Vidakovic] M. Vidakovic, N. Teslic, T. Maruna, V. Mihic, "*Android4TV: a proposal for integration of DTV in Android devices,*" 30th

IEEE International Conference on Consumer Electronics (ICCE), January 2012, pp. 441-442.

[Vidakovic2] M. Vidakovic, T. Maruna, N. Teslic, V. Mihic, "*A java API interface for the integration of DTV services in embedded multimedia devices,*" Consumer Electronics, IEEE Transactions on, vol.58, no.3, pp.1063,1069, August 2012.

[WIPO] *World Intellectual Property Organization* - WIPO, www.wipo.int

[Wagner] R.A. Wagner and M.J. Fischer, "*The string-to-string correction problem,*" J. ACM, vol. 21, no. 1, pp. 168-173, 1974.

[Webkit] The WebKit Open Source Project, www.webkit.org/

[Yaghmour] K. Yaghmour, "*Embedded Android: Porting, Extending, and Customizing,*" (1st ed.), O'Reilly Media, Inc., 2013.

DODATAK A:

SERVISNE INFORMACIJE (SI) DVB STANDARDA

Ovaj dodatak pruža pregled servisnih informacija (SI) prisutnih u svakoj DVB mreži, neophodnih za razumevanje tehničkih detalja i ograničenja servisa digitalne televizije, čija integracija je predmet istraživanja ove disertacije. Dodatak takođe uključuje i informacije o formatu tabela servisnih informacija, njihovom međusobnom odnosu, kao i o ograničenjima kojima utiču na prijemnike. Sastoji se od bitnih delova standarda ISO 13818-1 [MPEG-2] i ETSI EN 300 468 [ETSI2], čija su struktura i opis preuzeti iz [Morris].

Transportni tok (engl. **T**ransport **S**tream, TS) sadrži dva različita tipa servisnih informacija. Prvi je definisan MPEG standardom i prijemnik ga koristi za pronalaženje tokova (engl. stream) koje treba dekodovati. Ovaj tip servisnih informacija predstavlja informacije specifične za program (engl. **P**rogram-**S**pecific **I**nformation, PSI). Drugi tip je takođe definisan standardom, a najčešće ili DVB ili ATSC standardima. On prijemniku pruža više podataka o drugim transportnim tokovima u mreži, a sadrži i dodatne podatke o transportnom toku koji su namenjeni korisnicima. DVB standard ovo naziva DVB servisnim informacijama (engl. DVB **S**ervice **I**nformation, DVB SI), dok ATSC ovo zove protokolom programskih i sistemskih informacija (engl. **P**rogram and **S**ystem **I**nformation **P**rotocol, PSIP). Oba standarda su zasnovana na formatima niskog nivoa, koje definiše MPEG, ali su informacije koje prenose različite.

Organizacija SI

PSI, DVB SI i PSIP se mogu posmatrati kao relacione baze podataka. Svaka se sastoji od skupa tabela koje sadrže podatke o transportnom toku i programima u okviru njega. Svaka tabela sadrži jedan tip podataka o transportnom toku, o servisima i elementarnim tokovima (engl. **E**lementary **S**trem, ES) koje on sadrži ili njegovim odnosima sa drugim transportnim tokovima u mreži. Zbog toga što svaka tabela koristi osnovnu strukturu definisanu specifikacijom [MPEG-2], male su razlike između različitih standarda o servisnim informacijama. Razlike su veće u strukturama tabela, nego u njihovim formatima.

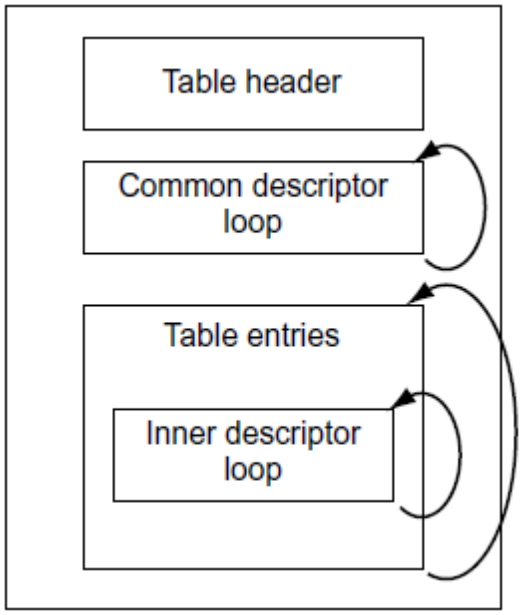
Deskriptori

Veliki deo podataka iz SI se prenosi kao skup deskriptora. Deskriptor je osnovna jedinica podataka koja odgovara delu reda u tabeli. Postoji mnogo različitih tipova deskriptora, od kojih svaki pruža jedan deo informacije iz baze podataka SI. Dobre osobine upotrebe deskriptora su:

- ***Prostor***: Neki delovi tabele nisu obavezni pa emiter televizijskog signala mora da pošalje samo deskriptore koji opisuju trenutnu situaciju.
- ***Jednostavnost parsiranja***: Korišćenje deskriptora pojednostavljuje parsiranje sadržaja kako bi određeni deo sadržaja postao neobavezan. U ovom slučaju, prijemnik treba da parsira samo jedan skup deskriptora, sličnog formata.
- ***Fleksibilnost***: Mrežni operatori mogu da šalju deskriptore bilo kojim redom, bez uticaja na njihovo značenje.
- ***Mogućnost ponovnog korišćenja***: Ukoliko je isti tip podataka potrebno poslati na više lokacija ili do više tabela, može se koristiti odgovarajući postojeći deskriptor. Ovo olakšava posao proizvođačima prijemnika, koji moraju da pišu programe za parsiranje, proizvođačima opreme za emitovanje televizijskog signala, koji moraju da generišu SI, kao i ljudima koji definišu standarde koji na ovaj način lakše održavaju konzistentnost.
- ***Proširivost***: Ukoliko je potrebno dodati podatke o novoj situaciji ili izmeniti sadržaj određenog dela podataka, dovoljno je dodati novi

deskriptor. Prijemnici koji ne podržavaju novi deskriptor će ga jednostavno preskočiti, dok će oni koji ga podržavaju moći da ga pročitaju i razumeju.

Deskriptori se obično šalju u skupovima poznatim kao petlje. Svaka tabela najčešće sadrži jednu zajedničku petlju, koja sadrži sve deskriptore koji se odnose na sve elemente tabele, i drugu petlju koja sadrži deskriptore koji čine redove tabela. Za ovu petlju, svaka iteracija će sadržati deskriptore iz jednog reda, ali može sadržati i ugnežđeni skup petlji u zavisnosti od toga koji deskriptori čine jedan red. Slika A.1, na primeru jedne tipične SI tabele, prikazuje kako ovakva struktura radi u bilo kom sistemu.



Slika A.1 Tipična SI tabela (preuzeto iz [Morris])

Svaki deskriptor ima svoju oznaku kako bi prijemnik mogao ispravno da ga parsira. Drugo polje zajedničko za sve deskriptore čuva veličinu deskriptora, kako bi prijemnik mogao da preskoči deskriptore čije oznake ne prepoznaje. Prilikom definisanja standarda za određeni deskriptor, opisuju se i sva ostala polja koja deskriptor sadrži, a prijemnici koji ne prepoznaju određeni deskriptor će ga samo preskočiti.

Neki od deskriptora, koji se najčešće pojavljuju u različitim SI tabelama, detaljnije će biti opisani u nastavku ovog dodatka. I pored toga što specifikacija DVB

SI definiše ciljnu lokaciju mnogih deskriptora, ne ograničava određeni deskriptor na ove lokacije, a u mnogim slučajevima ne definiše koji su deskriptori obavezni. Dokument sa preporukama za realizaciju DVB SI - [ETSI8] sadrži više detalja o ovome pa je važan izvor informacija za svakoga ko je zainteresovan za DVB SI. Preporučljivo je pogledati i dokument i specifikaciju DVB SI - [ETSI2] kako bi se osigurao pristup najnovijim informacijama, kao i da bi se imao detaljan uvid u opis i ograničenja DVB SI.

Prenos SI tabele

Prilikom prenosa, svaka SI tabela se deli na više paketa, tj. sekcija. Najveći broj paketa sa SI podacima koristi format privatne sekcije MPEG-2 standarda. Više detalja se može naći i u prvom delu specifikacije MPEG-2 standarda. Kada se tabela podeli u sekcije, one se prenose kao deo osnovnog MPEG toka, a cela tabela se emituje u pravilnim razmacima, kako bi prijemnik sigurno dobio celu kopiju tabele.

Kako bi prijemnik sigurno znao gde da nađe ove sekcije, neke tabele se emituju na dobro poznatim PID (engl. **P**acket **ID**entifirer) vrednostima. Ove tabele se mogu razlikovati između standarda, ali prijemniku govore gde se nalaze ostale tabele koje su mu potrebne. Prijemnik nadgleda ove poznate PID-ove, preuzima tabele, i analizira ih da bi utvrdio koje druge PID-ove treba da nadgleda kako bi učitao sve SI koje se emituju.

Ipak, prijemnik mora da nastavi nadgledanje ovih PID-ova jer mrežni operater u bilo kom trenutku može da ažurira sadržaj tabela. Svaka tabela sadrži broj verzije koji se menja kad god mrežni operater izmeni datu tabelu. Kada prijemnik detektuje da je verzija tabele promenjena, briše sve sačuvane podatke iz originalne tabele, i menja ih novim.

Informacije specifične za program

Ova sekcija se bavi sadržajem tabela. Pošto je PSI zajednički i za DVB i za ATSC, prvo će on biti obrađen. Transportni tok može nositi više od jednog programa (tj. više od jedne grupe povezanih tokova audio, video sadržaja). U okviru DTV-a, program se još naziva i servis ili kanal. Da bi utvrdio koji se programi nalaze u okviru jednog transportnog toka, prijemnik ispituje PAT tabelu (engl. **P**rogram **A**ssociation **T**able), čiji format prikazuje Tabela A.1 (a identifikacioni broj tabele je 0x00). Ovde

su nabrojani svi programi sadržani u transportnom toku kao i podaci koji prijemniku govore gde se nalaze detaljne informacije o svakom od ovih programa. PAT tabela ne predstavlja ništa drugo nego sadržaj transportnog toka.

Svako polje u PAT tabeli daje PID koji sadrži PMT tabelu (engl. **P**rogram **M**ap **T**able) tog programa. PMT je nešto detaljniji opis programa koji prijemniku govori koji PID-ovi su deo tog programa i koje tipove podataka sadrže. Ne sadrži informacije korisne i namenjene za korisnika. Prijemniku je ovo potrebno kako bi odredio koje tokove treba da dekoduje za određeni program. Format PMT tabele prikazuje Tabela A.2, a svaka PMT tabela ima identifikacioni broj tabele 0x02.

PID koji pripada jednom elementarnom toku (najčešće video ili audio elementarni tok koji sadrži kompresovane video ili audio podatke) može se naći u više PMT tabela i u tom slučaju se deli između svih programa koji se odnose na taj tok. Bez obzira na to koliko se programa odnosi na njega, potrebna je samo jedna instanca elementarnog toka u transportnom toku.

Sintaksa	*Broj bita*	*Identifikator*
program_association_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
'0'	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
transport_stream_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
for(i=0; i<N; i++){		
program_number	16	uimsbf
reserved	3	bslbf
if(program_number=='0') {		
network_PID	13	uimsbf
}		
else{		
program_map_PID	13	uimsbf
}		
}		
CRC_32	32	rpchof
}		

Tabela A.1 Format PAT tabele (izvor [MPEG-2])

Sintaksa	*Broj bita*	*Identifikator*
TS_program_map_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
'0'	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
program_number	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
reserved	3	bslbf
PCR_PID	13	uimsbf
reserved	4	bslbf
program_info_length	12	uimsbf
for(i=0; i<N; i++){		
descriptor()		
}		
for(i=0; i<N1; i++){		
stream_type	8	uimsbf
reserved	3	bslbf
elementary_PID	13	uimsnf
reserved	4	bslbf
ES_info_length	12	uimsbf
for(i=0; i<N2; i++){		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

Tabela A.2 Format PMT tabele (izvor [MPEG-2])

Informacije uslovnog pristupa

U bilo kom transportnom toku, moguće je enkriptovati jedan ili više elementarnih tokova kako bi bili dostupni samo onim korisnicima u mreži koji su pretplaćeni na njih. Postoji veliki broj različitih sistema za uslovni pristup pa je prijemniku potreban način na koji će utvrditi koji od tih sistema se koristi za određeni tok. Ovaj podatak se čuva u tabeli uslovnog pristupa (engl. **C**onditional **A**ccess **T**able, CAT), koju ukratko opisuje Tabela A.3. Identifikacioni broj ove tabele je 0x01.

Petlja deskriptora u CAT tabeli sadrži 0 ili više deskriptora uslovnog pristupa. Deskriptor uslovnog pristupa je opisan specifikacijom MPEG-2, i svaki deskriptor uslovnog pristupa prijemniku govori o jednom sistemu uslovnog pristupa korišćenom

u tom transportnom toku, kao i o tome koji PID se koristi za prenos poruka vezanih za taj CA sistem. CA deskriptori u CAT tabeli identifikuju PID-ove koji sadrže poruke koje se odnose na ceo sistem, kao na primer EMM poruke za kontrolu prava pristupa (engl. **E**ntitlement **M**anagement **M**essages), dok CA deskriptori u PMT tabeli identifikuju poruke poput EMC kontrolne poruke (engl. **E**ntitlement **C**ontrol **M**essages), koje se odnose samo na taj program.

Sintaksa	*Broj bita*	*Identifikator*
`A_section(){`		
`  table_id`	`8`	`uimsbf`
`  section_syntax_indicator`	`1`	`bslbf`
`  '0'`	`1`	`bslbf`
`  reserved`	`2`	`bslbf`
`  section_length`	`12`	`uimsbf`
`  reserved`	`18`	`bslbf`
`  version_number`	`5`	`uimsbf`
`  current_next_indicator`	`1`	`bslbf`
`  section_number`	`8`	`uimsbf`
`  last_section_number`	`8`	`uimsbf`
`  for(i=0; i<N; i++){`		
`    descriptor()`		
`  }`		
`  CRC_32`	`32`	`rpchof`
`}`		

Tabela A.3 Format CAT tabele (izvor [MPEG-2])

DVB Servisne informacije (SI)

U nastavku se analiziraju DVB Servisne informacije (SI). Potpuni opisi se nalaze u okviru dokumentacije odgovarajućeg standarda. Tabela A.4 navodi sve standarde koji sadrže neke informacije o SI u DVB mrežama.

Pronalaženje informacija o mreži

PSI tabele su korisne za opisivanje sadržaja jednog transportnog toka, ali se mreže najčešće sastoje od više transportnih tokova. Ovo znači da prijemnik mora imati mehanizam za otkrivanje transportnih tokova u jednoj mreži, kao i načina na koje im može pristupiti. U DVB sistemima, ovi podaci se nalaze u tabelama koje čuvaju informacije o mreži (engl. **N**etwork **I**nformation **T**able, NIT).

Standard	*Opis*
TR 101 154	Implementation guidelines for MPEG in broadcasting applications.
EN 300 468	DVB Service Information (DVB-SI) specification.
TR 101 211	Implementation guidelines for DVB-SI.
EN 301 192	DVB Data Broadcasting specification. Two versions of this specification are of interest to MHP developers. Version 1.2.1 applies to version 1.0.2 and earlier of MHP, as well as MHP 1.1 and GEM. Version 1.3.1 applies to MHP version 1.0.3 and MHP 1.1.1.
TR 101 202	Implementation guidelines for DVB data broadcasting.
TR 101 162 (ETR 162)	Allocation of SI codes for DVB systems.

Tabela A.4 SI standardi za DVB sisteme (preuzeto iz [Morris])

NIT čuva listu transportnih tokova u mreži i parametre koji su potrebni prijemniku da bi se podesio za njih. U nekim slučajevima, može biti podeljena u dve podtabele: NIT za trenutnu mrežu (NIT-actual) i NIT za sve druge mreže koje prijemnik može da primi (NIT-other). Svaka mreža mora imati NIT-actual tabelu, dok je NIT-other proizvoljna i potrebna samo u nekim slučajevima.

Pošto je NIT toliko važna prijemniku, uvek se prenosi na istom PID-u (0x10) u okviru transportnog toka. NIT-actual i NIT-other se prenose na istom PID-u, ali imaju različite identifikacione brojeve kako bi prijemnik mogao da ih razlikuje. Identifikacioni broj tabele za sekcije koje sadrže NIT-actual je 0x40, dok je za sekcije koje sadrže NIT-other 0x41. Svaka sekcija koja sadrži NIT ima strukturu koju prikazuje Tabela A.5.

Svaka mreža ima jedinstveni identifikacioni broj po kom je prijemnik razlikuje od ostalih. Takođe, u okviru mreže, svaki transportni tok ima svoj identifikacioni broj, iako jedna mreža može dozvoliti drugim mrežama da prenose jedan ili više njenih transportnih tokova. U tom slučaju, može se desiti da dve mreže prenose isti transportni tok sa jednim identifikacionim brojem toka, ali različitim identifikacionim brojevima mreža. Kako bi DTV prijemnik lakše identifikovao ovaj slučaj, NIT tabele takođe u sebi imaju identifikacioni broj mreže koja je prvobitno generisala taj sadržaj. Ovo se naziva identifikacionim brojem originalne mreže. Zajedno, identifikacioni broj originalne mreže i identifikacioni broj transportnog toka (koji se čuvaju u NIT tabeli)

na jedinstven način definišu određeni transportni tok, bez obzira na to koja ga mreža trenutno prenosi.

Prva deskriptorska petlja NIT tabele mora da uključuje deskriptor imena mreže, dok su ostali deskriptori, u najvećem broju slučajeva, opcioni. Unutrašnja deskriptorska petlja, koja opisuje individualne transportne tokove, mora uključivati i deskriptor koji identifikuje sistem isporuke (deskriptor satelitskog, kablovskog ili zemaljskog sistema isporuke). Osim tih, i drugi deskriptori mogu biti prisutni, ali nisu obavezni.

Sintaksa	*Broj bita*	*Identifikator*
network_information_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
network_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
reserved_future_use	4	bslbf
network_descriptors_length	12	uimsbf
for(i=0; i<N; i++){		
descriptor()		
}		
reserved_future_use	4	bslbf
transport_stream_loop_length	12	uimsbf
for(i=0; i<N; i++){		
transport_stream_id	16	uimsbf
original_network_id	16	uimsbf
reserved_future_use	4	bslbf
transport_descriptors_length	12	uimsbf
for(j=0; j<N; j++){		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

Tabela A.5 Format NIT tabele (izvor [ETSI2])

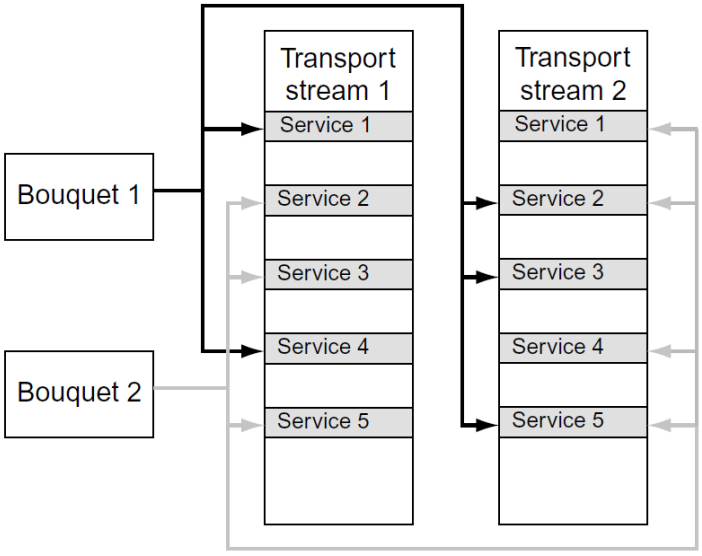
Kolekcije

Transportni tokovi organizuju tokove u mreži na fizičkom nivou. Svaki servis pripada jednom transportnom toku, a svaki transportni tok se emituje na različitoj frekvenciji sa određenim skupom promenljivih parametara.

Ipak, mrežni operateri ponekad svoje servise žele da organizuju na drugačiji način. Nekada žele da grupišu kanale po tematici (na primer, jedna grupa za sportske kanale, druga za kanale sa informativnim vestima, treća za filmske kanale, …), ili da ih grupišu po pretplatničkim paketima. Na primer, svi servisi u osnovnom paketu mogu biti u jednoj grupi, dok druga grupa može sadržati servise koji čine „filmski“ paket, a neka treća, servise namenjene deci. Još komplikovaniji slučaj bi bio onaj u kom mrežni operater neke kanale stavlja u više servisnih grupa.

Potreban je neki način za grupisanje servisa, bez obzira na transportne tokove. DVB koristi koncept „kolekcija“ (engl. bouquet) da na takav način grupiše servise. Kolekcija (Slika A.2) je skup servisa povezanih logički, bez obzira na transportni tok kom pripadaju. Kolekcije mogu grupisati čak i servise iz više mreža.

Kolekcije su opisane u neobaveznoj tabeli asocijacije kolekcija (engl. **B**ouquet **A**ssociation **T**able, BAT). Mrežni operateri ovu tabelu mogu da koriste za identifikaciju kolekcija prisutnih u transportnom toku, kao i za određivanje servisa tog transportnog toka koji su delovi tih kolekcija. Tabela A.6 prikazuje strukturu BAT tabela.



Slika A.2 Kolekcije su logičke grupe servisa koje mogu da dolaze iz različitih transportnih tokova (preuzeto iz [Morris])

Sintaksa	*Broj bita*	*Identifikator*
bouquet_association_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
Reserved	2	bslbf
section_length	12	uimsbf
bouquet_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
reserved_future_use	4	bslbf
bouquet_descriptors_length	12	uimsbf
for(i=0; i<N; i++){		
descriptor()		
}		
reserved_future_use	4	bslbf
transport_stream_loop_length	12	uimsbf
for(i=0; i<N; i++){		
transport_stream_id	16	uimsbf
original_network_id	16	uimsbf
reserved_future_use	4	bslbf
transport_descriptors_length	12	uimsbf
for(j=0; j<N; j++){		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

Tabela A.6 Format BAT tabele (izvor [ETSI2])

BAT sekcije imaju identifikacioni broj tabele 0x4A i podeljene su u nekoliko podtabela koje opisuju po jednu kolekciju. Svako polje u podtabeli se odnosi na jedan transportni tok i sadrži skup deskriptora, među kojima obavezno mora biti deskriptor liste servisa koji identifikuje sve servise iz tog transportnog toka, a koji su deo kolekcije. Servis može biti deo više od jedne kolekcije, pa se samim tim jedan servis može pojaviti u više BAT podtabela.

Svaka kolekcija ima ime, koje se definiše ili deskriptorom imena kolekcije, ili višejezičnim deskriptorom imena kolekcije, koji se prenosi u spoljašnjoj petlji. Spoljašnja deskriptorska petlja može sadržati i do dva deskriptora dostupnosti na nivou država za svaku BAT podtabelu. Ovo omogućava mrežnom operateru da identifikuje određene zemlje koje treba da prime kolekciju, i određene koje ne treba. Druga

deskriptorska petlja mora sadržati deskriptor liste servisa. DVB će preskočiti sve druge deskriptore u ovoj petlji.

Opisivanje servisa u DVB

Pomoću NIT i BAT tabela prijemnik zna kako su transportni tokovi organizovani unutar mreže, ali to nije ni od kakve koristi gledaocu, kome je mnogo bitnije koje kanale ima na raspolaganju. PAT i PMT govore prijemniku šta se nalazi unutar toka, ali ni to nije od koristi gledaocu jer se one fokusiraju na podatke koje prijemnik treba da zna, a ne na one koje gledalac želi da zna.

Kako bi pružili više informacija korisniku, DVB sistemi koriste tabelu opisa servisa (engl. **S**ervice **D**escription **T**able, SDT). Ona sadrži informacije o toku i programima unutar njega, kao i imena različitih servisa, informacije o roditeljskoj blokadi, kao i tekstualni opis servisa. Strukturu SDT prikazuje Tabela A.7.

SDT može biti podeljena u dve podtabele, kako bi se prijemniku i korisniku olakšalo pronalaženje dostupnih servisa. Prva podtabela opisuje servise koji su dostupni u okviru trenutnog transportnog toka, naziva se SDT-actual i ima identifikacioni broj 0x42. Druga podtabela je SDT-other i opisuje koji su servisi dostupni u drugim transportnim tokovima u mreži. Njen identifikacioni broj je 0x46.

Tabela SDT-other nije obavezna pa se ne mora nalaziti u svakom transportnom toku. Mrežni operateri je mogu koristiti za obaveštavanje prijemnika o svim servisima u mreži, a da ih ne teraju da se zakače na svaki transportni tok kako bi očitali njegovu SDT-actual tabelu. Iako prijemnici ovo moraju da urade samo prilikom početka rada, kako bi napravili listu dostupnih servisa, proces može biti spor. Korišćenje SDT-other tabele znači da prijemnik ne mora da čuva podatke o servisima između restartovanja, ili da ih traži svakog puta kad mu zatrebaju.

Svaki unos u SDT tabeli ima jedan par oznaka (EIT_schedule_flag i EIT_present_following_flag), koje prijemniku govore koje informacije o događaju su prisutne, tako da može da izbegne traženje nepostojećih tabela. Servisi imaju i aktivan status koji označava da li se određeni servis trenutno emituje.

Svako polje SDT tabele ima deskriptorsku petlju, koja mora uključivati deskriptor servisa. Ovo pruža osnovne informacije o servisu, kao što su ime i tip servisa (na primer DTV servis, radio servis, ili servis koji uključuje opšte podatke).

Elektronski programski vodič ovo može koristiti da gledaocu ponudi više informacija o dostupnim servisima.

Još jedan važan deskriptor koji može biti prisutan je deskriptor emitovanja podataka. Ovaj deskriptor je najkorisniji DTV MW programskoj podršci prijemnika i definiše tokove podataka u servisu. Takođe, govori prijemniku koji tipovi tokova podataka su prisutni, i obaveštava ga o parametrima koji su mu potrebni da bi pristupio tom toku. Pored toga, može sadržati i tekstualni opis toka, kako bi prijemnik mogao korisniku da da malo više informacija o njemu. U zavisnosti od tipa servisa za emitovanje podataka, ovaj deskriptor može biti uključen u EIT, umesto u SDT tabelu. Ukoliko servis sadrži više od jednog toka podataka, polje tog servisa u SDT će sadržati po jedan deskriptor emitovanja podataka za svaki tok podataka. Kako bi prijemnici lakše našli SDT tabelu, svaki transportni tok prenosi SDT (i SDT-actual i SDT-other) na PID-u 0x11.

Sintaksa	*Broj bita*	*Identifikator*
service_description_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
transport_stream_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
original_network_id	16	uimsbf
reserved_future_use	8	bslbf
for(i=0; i<N; i++){		
service_id	16	uimsbf
reserved_future_use	6	bslbf
EIT_schedule_flag	1	bslbf
EIT_present_following_flag	1	bslbf
running_status	3	uimsbf
free_CA_mode	1	bslbf
descriptors_loop_length	12	uimsbf
for (j=0; j<N; j++){		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

Tabela A.7 Format SDT tabele (izvor [ETSI2])

Opisivanje događaja

SDT tabela korisnicima omogućava da pronađu kanale koje žele, ali im ništa ne govori o emisijama koje se puštaju na tim kanalima. Informacije o emisijama koje će biti prikazane u okviru pojedinih servisa čuva tabela informacija o događajima (engl. **E**vent **I**nformation **T**able).

EIT tabela može biti podeljena u nekoliko podtabela. Svaki transportni tok mora nositi informacije o trenutnom i narednom događaju, za svaki servis u tom transportnom toku. Ovo se prenosi u trenutnoj/narednoj EIT tabeli. Po potrebi, mogu biti dodate i informacije o događajima u budućnosti, kako bi EPG korisniku mogao prikazati raspored programa. Ovo se prenosi u EIT tabeli rasporeda.

Kao što je slučaj i sa SDT tabelom, EIT tabela može prenositi informacije i o događajima u drugim transportnim tokovima. I EIT trenutni/naredni događaj, i EIT raspored događaja iz ostalih transportnih tokova mogu biti prisutne, kao odvojene podtabele, kako EPG ne bi morao da se priključuje na drugi transportni tok da bi dobio sve podatke o rasporedu. Potpune informacije o rasporedima svih servisa u mreži mogu zauzimati mnogo mesta, pa emiteri najčešće prikazuju samo trenutne/naredne informacije o drugim transportnim tokovima.

Zbog količine prostora koju informacije o rasporedu mogu da zauzmu, DVB definiše kako bi ih trebalo prenositi u EIT tabeli. Svaki blok od osam sekcija u EIT tabeli rasporeda se naziva segment. Svaki segment predstavlja tročasovni period u rasporedu, i taj segment sadrži podatke o svakom događaju koji počinje u okviru tog perioda. Ovi podaci su raspoređeni hronološki u okviru segmenta, kao i segmenti u okviru tabele. Kako bi prijemniku bilo olakšano pronalaženje podataka za određeni vremenski period, prvi segment u tabeli sadrži podatke o rasporedu počevši od ponoći tekućeg dana. Prijemnici ovo mogu koristiti za određivanje segmenta koji treba da učitaju.

Svaka tabela može imati do 256 sekcija, koje predstavljaju informacije o rasporedu za četiri dana. Pošto ovo ne mora biti dovoljno nekim aplikacijama, informacije o rasporedu iz EIT tabele mogu biti podeljene na nekoliko podtabela. Ove podtabele koriste identifikacione brojeve tabela od 0x50 do 0x5f za emitovani, a od 0x60 do 0x6f za druge transportne tokove. Identifikacioni brojevi tabela se, takođe,

koriste hronološkim redom. To znači da će ID 0x50 biti korišćen za prva četiri sata željenog transportnog toka, 0x51 za naredna četiri sata i tako dalje.

Kompletne informacije o rasporedu bi zauzimale veliki deo protoka pa emiteri obično ne emituju sve podtabele, a najčešće prenose samo trenutne/naredne informacije ostalih transportnih tokova. Takođe, mogu odabrati da ne emituju informacije o rasporedu prošlih segmenata. U tom slučaju, segmenti i dalje moraju biti emitovani, ali bez pojedinačnih elemenata. Sve sekcije koje prenose EIT informacije imaju format koji opisuje Tabela A.8.

Sintaksa	*Broj bita*	*Identifikator*
event_information_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
service_id	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
transport_stream_id	16	uimsbf
original_network_id	16	uimsbf
segment_last_section_number	8	uimsbf
last_table_id	8	uimsbf
for(i=0;i<N;i++){		
event_id	16	uimsbf
start_time	40	bslbf
duration	24	uimsbf
running_status	3	uimsbf
free_CA_mode	1	bslbf
descriptors_loop_length	12	uimsbf
for(i=0;i<N;i++){		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

Tabela A.8 Format EIT tabele (izvor [ETSI2])

Svaka sekcija EIT tabele opisuje događaje samo jednog servisa. Različite sekcije mogu sadržati informacije o događajima različitih servisa jednog transportnog toka.

Svaki događaj ima identifikacioni broj koji ga na jedinstven način predstavlja u okviru servisa (iako brojevi događaja mogu biti korišćeni više puta u okviru transportnog toka). Pored vremena početka i dužine trajanja događaja, EIT prijemniku govori i da li je bilo koji deo događaja zaštićen (korišćenjem free_CA_mode oznake).

Početno vreme svakog događaja je predstavljeno parom vreme/datum. To polje je kodovano korišćenjem dve različite reprezentacije, pa je o njemu lakše razmišljati kao o dva odvojena polja koja su spojena u jedno. Najznačajnijih 16 bita polja sadrže datum (korišćenjem modifikovane julijanske reprezentacije datuma, date brojem dana proteklih od 17. novembra 1858). Najmanje značajnih 24 bita ovog polja predstavljaju vreme u toku dana kad događaj počinje. Ovaj deo se koduje 4-bitnim BCD kodovanjem, kako bi svaki bajt predstavljao sate, minute ili sekunde vremena početka (u formi 24 sata u danu). Takvim kodovanjem, početno vreme 10:45 p.m. bi bilo čuvano u najmanje značajnih 24 bita kao 0x224500. Trajanje je, takođe, kodovano 4-bitnim BCD kodovanjem. U ovom slučaju, nisu potrebne informacije o datumu jer događaji retko traju duže od jednog celog dana.

Pošto događaji nekad ne počnu kad bi trebalo, svako polje EIT tabele sadrži i running_status polje. Ovo prijemniku govori o statusu događaja iz EIT trenutne/naredne tabele događaja i može biti korišćeno za rešavanje situacija u kojima je zakazano početno vreme događaja proteklo, a događaj još nije počeo, ili je počeo, ali je bio prekinut.

Opis svakog događaja mora sadržati jedan ili više deskriptora komponente u deskriptorskoj petlji. Oni opisuju audio ili video tokove koji su deo događaja. Tokovi podataka koriste deskriptore emitovanja podataka, umesto deskriptora komponenti, i u ovom slučaju, jedan deskriptor emitovanja podataka treba da bude prisutan za svaki tok podataka u događaju. Ukoliko svaki događaj u servisu koristi iste tokove podataka, SDT može, umesto EIT tabele, sadržati deskriptor emitovanja podataka.

Pored deskriptora komponenti, svaki opis događaja uključuje i kratak deskriptor događaja. On sadrži ime događaja, a nekad i kratak tekstualni opis. U slučaju da ovo nije dovoljno, mrežni operater može uključiti i prošireni deskriptor događaja koji, kako mu ime kaže, sadrži duži opis datog događaja.

Ukoliko se za mnogo događaja istovremeno ažurira status, mrežni operater može ove informacije emitovati zasebno, putem tabele statusa (engl. **R**unning **S**tatus **T**able,

RST). Ovo mrežnom operateru omogućava da ažurira statuse jednog ili više događaja bez ponovnog emitovanja cele EIT tabele događaja. Status definisan u RST tabeli ima veći značaj od onog koji je emitovan u EIT tabeli događaja, ali EIT tabela mora biti ažurirana ispravnim statusom pre nego što se opet pošalje. Sekcije koje sadrže RST imaju identifikacioni broj 0x71 i format opisuje Tabela A.9.

Sintaksa	*Broj bita*	*Identifikator*
running_status_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
for (i=0; i<N; i++){		
transport_stream_id	16	uimsbf
original_network_id	16	uimsbf
service_id	16	uimsbf
event_id	16	uimsbf
reserved_future_use	5	bslbf
running_status	3	uimsbf
}		
}		

Tabela A.9 Format RST tabele (izvor [ETSI2])

Određivanje vremena

Podaci o vremenu početka nekog događaja su korisni samo ako je časovnik prijemnika sinhronizovan sa časovnikom mrežnog operatera. Kako bi se ovo osiguralo, mrežni operater emituje tabelu vremena i datuma (engl. **T**ime and **D**ate **T**able, TDT) koja prijemniku govori trenutno vreme. Pošto je TDT toliko mala, prenosi se jednom MPEG-2 sekcijom, koja ima format koji prikazuje Tabela A.10. Identifikacioni broj TDT tabele je 0x70.

Vreme je predstavljeno kao trenutno UTC vreme, kodirano u istom formatu korišćenom za početna vremena u EIT tabelama. Naravno, UTC ne koristi svaka zemlja koja koristi DVB sisteme, pa DVB definiše drugu tabelu, koja označava razliku koju bi prijemnik trebalo da uračuna kako bi dobio ispravno lokalno vreme. Ova tabela se zove tabela vremenske razlike (engl. **T**ime **O**ffset **T**able, TOT). Kao i TDT, i ova

tabela je sadržana u jednoj MPEG-2 sekciji, kao što prikazuje Tabela A.11, a ima identifikacioni broj 0x73. I TDT i TOT tabele su neobavezne.

Sintaksa	*Broj bita*	*Identifikator*
time_date_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
UTC_time	40	bslbf
}		

Tabela A.10 Format TDT tabele (izvor [ETSI2])

Sintaksa	*Broj bita*	*Identifikator*
time_offset_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
reserved_future_use	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
UTC_time	40	bslbf
reserved	4	bslbf
descriptors_loop_length	12	uimsbf
for(i=0; i<N; i++){		
descriptor()		
}		
CRC_32	32	rpchof
}		

Tabela A.11 Format TOT tabele (izvor [ETSI2])

Kao što je slučaj i sa TDT tabelom, i ova tabela čuva vreme u UTC formi. Međutim, pored toga, u deskriptorskoj petlji može sadržati i deskriptor razlike. Ovaj deskriptor definiše razliku između UTC i lokalne predstave vremena za različite zemlje i ima format koji prikazuje Tabela A.12.

Deskriptor razlike između UTC i lokalnog vremena može opisivati razliku između vremenskih zona za više zemalja, a svaka zemlja je predstavljena svojom standardnom, ISO, skraćenicom od tri slova (npr. FRA za Francusku). Takođe, svaka zemlja ima i regionalnu identifikaciju koja se koristi za razlikovanje različitih

vremenskih zona u jednoj zemlji. Na primer, svi podaci za Sjedinjene Američke Države bi imali kod zemlje USA, ali različite regionalne kodove za EST, CST i druge vremenske zone.

Sintaksa	*Broj bita*	*Identifikator*
`local_time_offset_descriptor(){`		
`  descriptor_tag`	`8`	`uimsbf`
`  descriptor_length`	`8`	`uimsbf`
`  for(i=0; i<N; i++){`		
`  country_code`	`24`	`bslbf`
`  country_region_id`	`6`	`bslbf`
`  reserved`	`1`	`bslbf`
`  local_time_offset_polarity`	`1`	`bslbf`
`  local_time_offset`	`16`	`bslbf`
`  time_of_change`	`40`	`bslbf`
`  next_time_offset`	`16`	`bslbf`
`  }`		
`}`		

Tabela A.12 Format deskriptora lokalne predstave vremena (izvor [ETSI2])

Za zemlje sa samo jednom vremenskom zonom, regionalna identifikacija je uvek nula, ali za zemlje sa više njih, regionalni ID brojevi počinju od 0x01 za najistočniju vremensku zonu i uvećavaju se za jedan za svaku zonu zapadno od te. Ako se uzme USA kao primer, atlantska vremenska zona će imati regionalnu identifikaciju 0x01, istočna vremenska zona 0x02, centralna 0x03, i tako dalje.

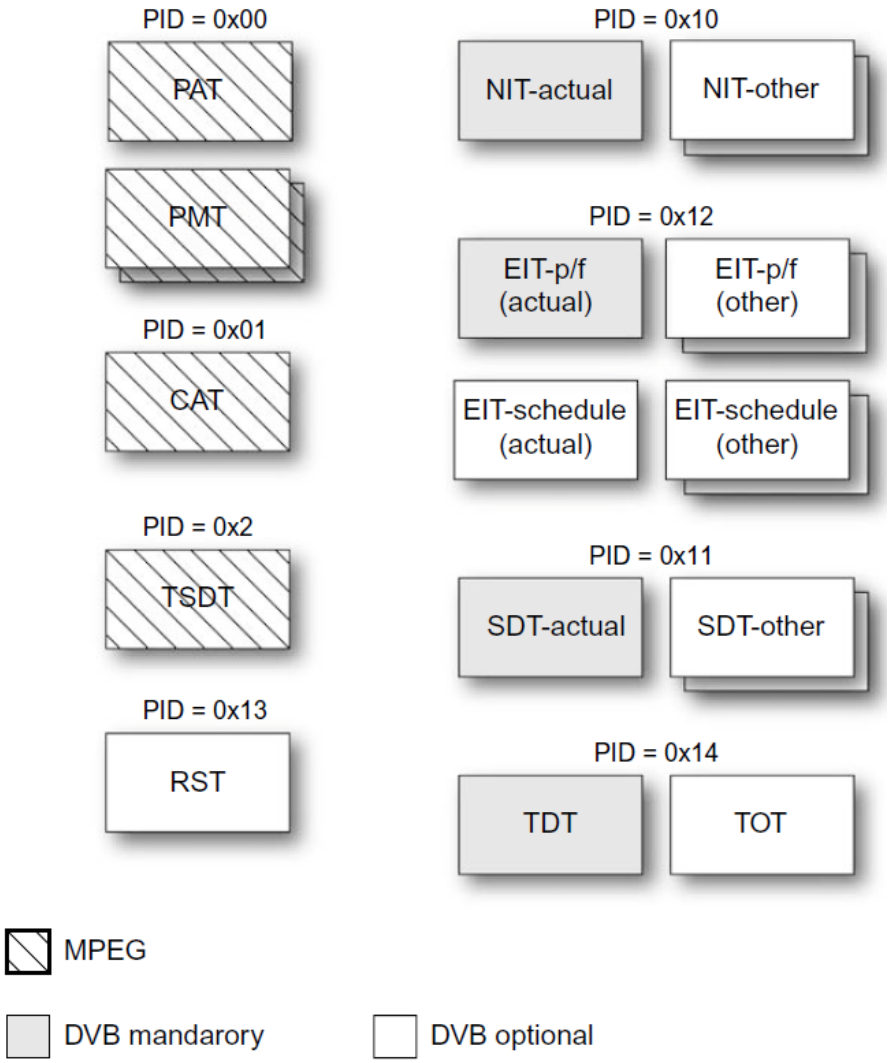
Lokalnu vremensku razliku određuju dva polja: polarnost, koja određuje da li je razlika pozitivna ili negativna u odnosu na UTC, i samo odstojanje (koje određuje razliku između lokalnog i UTC vremena). Razlika se čuva kao vrednost kodovana 4-bitnim BCD kodom u formatu HHMM, kako bi odstojanje od +8 sati bilo kodovano kao 0x0800. razlika je uvek u rasponu od -12 do +13 sati.

Zahvaljujući početku ili kraju letnjeg računanja vremena, ove razlike se povremeno mogu menjati. Kako bi se ovo podržalo, deskriptor razlike između lokalnog i UTC vremena prijemniku govori kada počinje, ili se završava, letnje računanje vremena, i koja je nova razlika. Sva vremena (nasuprot razlikama) u deskriptoru razlike između lokalnog i UTC vremena predstavljena su u istom formatu kao i početna vremena događaja u EIT tabelama.

Sažetak

Prethodno opisano predstavlja većinu tabela koje čine DVB-SI. Postoji još nekoliko koje nisu uključene, ali one uglavnom nisu neophodne za razumevanje koncepata DVB-SI. Slika A.3 prikazuje međusobni odnos različitih SI tabela.

Svaka tabela se prenosi uz ponavljanja, kako bi se osiguralo da će ih prijemnik primiti u celosti. Učestanost ponavljanja ovih tabela određuje mrežni operater, ali je minimalna učestanost za svaku tabelu određena DVB standardom. O ovome će biti detaljnije diskutovano u nastavku.



Slika A.3 Odnos između različitih SI tabela (preuzeto iz [Morris])

Tabela A.13 prikazuje PID-ove koji se koriste za emitovanje različitih SI tabela, kao i koliko često te tabele treba emitovati. Treba napomenuti da su ovo minimalne učestanosti ponavljanja i da mrežni operater može izabrati da ih emituje češće, ukoliko to poboljšava funkcionisanje mreže. Iako su nabrojane minimalne učestanosti svake tabele, emitovanje SI je više od toga. DVB ukazuje na to da bi pauza između slanja sekcija iste tabele trebalo da bude manja od 25 ms, što znači da bi mrežni operateri trebalo da povećaju učestanost slanja u zavisnosti od veličine SI tabela.

Tabela	*PID*	*Min. period ponavljanja*
PAT	0x0000	100 milliseconds
PMT	def. u PAT	100 milliseconds
CAT	0x0001	Not specified
NIT	0x0010	10 seconds
BAT	0x0011	10 seconds
SDT	0x0011	2 seconds
EIT	0x0012 0x0010 0x0011 0x0012 0x0013 0x0014	Tabela A.14 Not specified
RST	0x0013	Not specified
TOT	0x0014	30 seconds
TDT	0x0014	30 seconds
TSDT	0x0002	10 seconds
Discontinuity Information Table (DIT)	0x001E	Not specified
Selection Information Table (SIT)	0x001F	Not specified

Tabela A.13 Minimalni period ponavljanja DVB SI tabela (preuzeto iz [Morris])

Učestanost ponavljanja EIT tabela zavisi od tipa mreže i informacija o događajima koje mreža prenosi. Tabela A.14 prikazuje različite učestanosti koje se mogu primeniti.

Kao i PID-ovi koje prikazuje Tabela A.13, DVB rezerviše i nekoliko drugih PID-ova. Mrežni operateri bi trebalo da izbegavaju korišćenje PID-ova koje koriste ATSC PSIP [Morris]. Ovo olakšava deljenje sadržaja između ATSC i DVB mreža i, iako ovo u mnogim slučajevima nije smetnja, nije preporučljiva praksa.

EIT podtabela	*Per. ponavljanja (sec)*
Satellite or Cable Networks	
EIT present/following (actual)	2
EIT present/following (other)	10
EIT schedule (events in the next 8 days)	10
EIT schedule (other events)	30
Terrestrial Networks	
EIT present/following (actual)	2
EIT present/following (other)	20
EIT schedule (actual, events in the first full day)	10
EIT schedule (other, events in the first full day)	60
EIT schedule (actual, all events)	30
EIT schedule (other, all events)	300

Tabela A.14 Minimalni period ponavljanja EIT podtabela (preuzeto iz [Morris])

Optimizacija iskorišćenja protoka: Tabela opisa transportnog toka

U nekim slučajevima, može postojati skup deskriptora koji se odnose na svaki servis u okviru transportnog toka. Iako bi se mogli uključiti u odgovarajuću BAT ili SDT tabelu, ovo bi značilo rasipanje dostupnog protoka, pa se stoga mogu uključiti u posebne tabele zvane tabele opisa transportnog toka (engl. **T**ransport **S**tream **D**escription **T**able, TSDT). Tabela A.15 prikazuje format sekcija koje prenose TSDT.

Sintaksa	*Broj bita*	*Identifikator*
TS_description_section(){		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
"0"	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
reserved	18	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
for(i=0; i<N; i++) {		
descriptor()		
}		
CRC_32	32	rpchof
}		

Tabela A.15 Format TSDT tabele (izvor [MPEG-2])

Pošto ne mogu svi deskriptori biti preneti kroz TSDT, neki od njih se, odvojeno, prenose u svakom SDT ili BAT unosu. I pored toga, uobičajeno je da prijemnici prvo provere TSDT, pre parsiranja drugih tabela, kako bi parsiranje SI bilo što jednostavnije.

DODATAK B:

PREGLED DTV JAVA PROGRAMSKE SPREGE

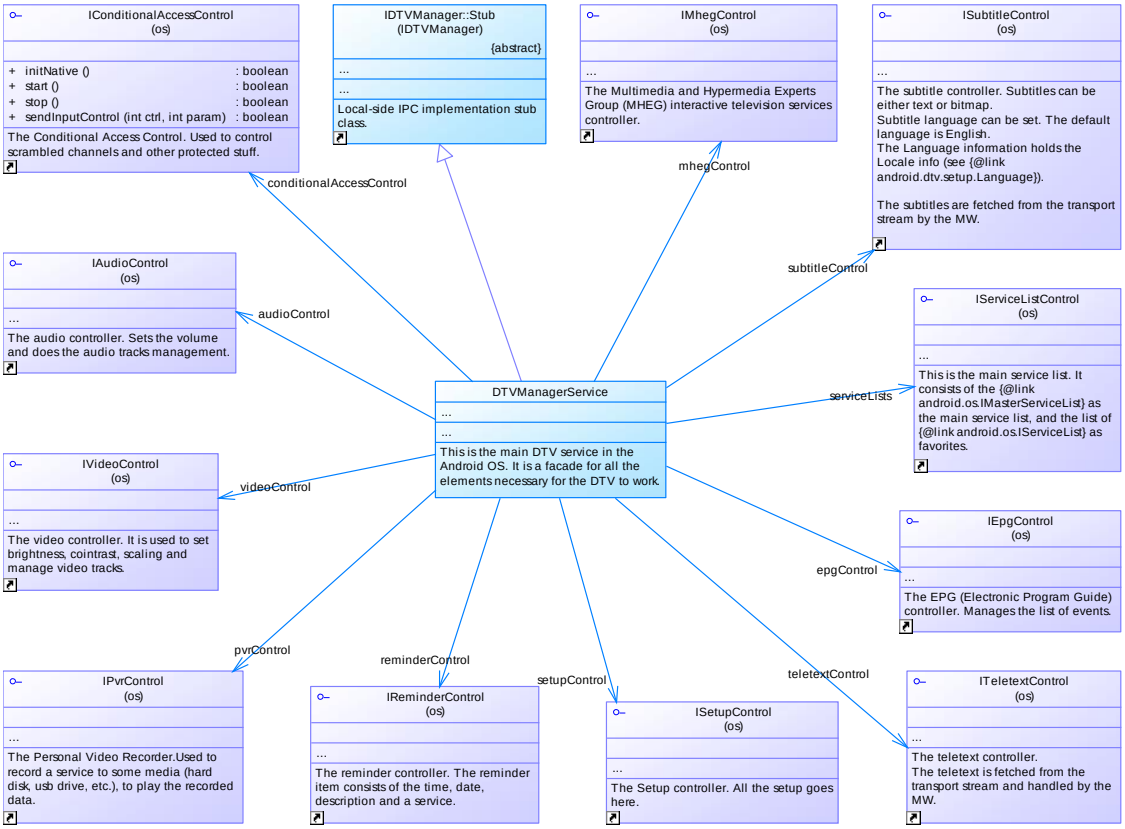
Ime paketa	Kratak opis
com.android.server	Obezbeđuje funkcionalnost DTV servisa implementiranu kroz DTVManagerService klasu. Ova klasa je vidljiva kroz *Binder* mehanizam.
android.dtv.audio	Obezbeđuje klase neophodne za kontrolu audio podataka u TV aplikaciji.
android.dtv.ca	Obezbeđuje klase neophodne za upravljanje kontrolisanim pristupom (CA) u TV aplikaciji.
android.dtv.callbacks	Obezbeđuje klase neophodne za internu razmenu asinhronih događaja.
android.dtv.epg	Obezbeđuje klase neophodne za upravljanje elektronskim programskim vodičem (EPG) u TV aplikaciji.
android.dtv.io	Obezbeđuje klase neophodne za kontrolu ulaznih/izlaznih sprega u TV aplikaciji.
android.dtv.mheg	Obezbeđuje klase neophodne za upravljanje MHEG interaktivnim TV servisima u TV aplikaciji.
android.dtv.ondemand	Obezbeđuje klase neophodne za upravljanje funkcionalnošću video na zahtev u TV aplikaciji.

Ime paketa	Kratak opis
android.dtv.parental	Obezbeđuje klase neophodne za upravljanje funcionalnošću roditeljske kontrole u TV aplikaciji.
android.dtv.picture	Obezbeđuje klase neophodne za upravljanje specijalnim podešavanjima izlazne slike u TV aplikaciji.
android.dtv.pvr	Obezbeđuje klase neophodne za upravljanjem personalnim video snimačem (PVR) u TV aplikaciji.
android.dtv.reminder	Obezbeđuje klase neophodne za upravljanje funkcionalnošću podsetnika u TV aplikaciji.
android.dtv.route	Obezbeđuje klase neophodne za upravljanje višestrukim rutama tokova podata u TV aplikaciji.
android.dtv.service	Obezbeđuje klase neophodne za upravljanje servisnim listama u TV aplikaciji.
android.dtv.setup	Obezbeđuje klase neophodne za konfigurisanje TV prijemnika iz TV aplikacije.
android.dtv.sound	Obezbeđuje klase neophodne za upravljanje podešavanjima izlaznog zvuka TV platforme iz TV aplikacije.
android.dtv.subtitle	Obezbeđuje klase neophodne za upravljanje prevodima iz TV aplikacije.
android.dtv.swupdate	Obezbeđuje klase neophodne za upravljanje mehanizmom ažuriranja programske podrške TV prijemnika.
android.dtv.teletext	Obezbeđuje klase neophodne za upravljanje funkcionalnošću teleteksta iz TV aplikacije.
android.dtv.utils	Obezbeđuje pomoćne klase neophodne za upravljanje podešavanjima datuma i vremena na TV prijemniku.
android.dtv.video	Obezbeđuje klase neophodne za upravljanje podešavanjima izlaznog video signala TV platforme iz TV aplikacije.
android.os	Obezbeđuje sprege neophodne za upravljanje TV aplikacijom.

Tabela B.1 Moduli DTV Java programske sprege (Android4TV)

Paket: com.android.server

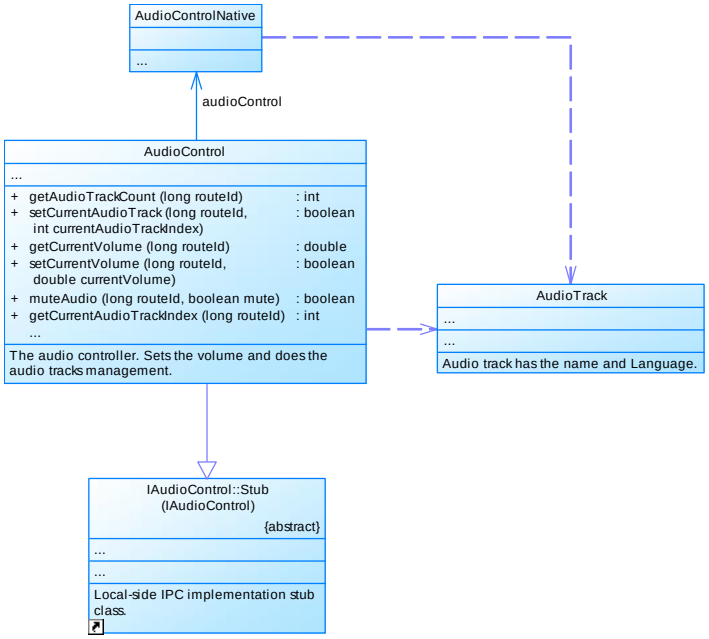
Obezbeđuje funkcionalnost DTV servisa implementiranu kroz DTVManagerService klasu. Ova klasa je vidljiva kroz *Binder* mehanizam.



Slika B.1 Sadržaj paketa com.android.server

Paket: android.dtv.audio

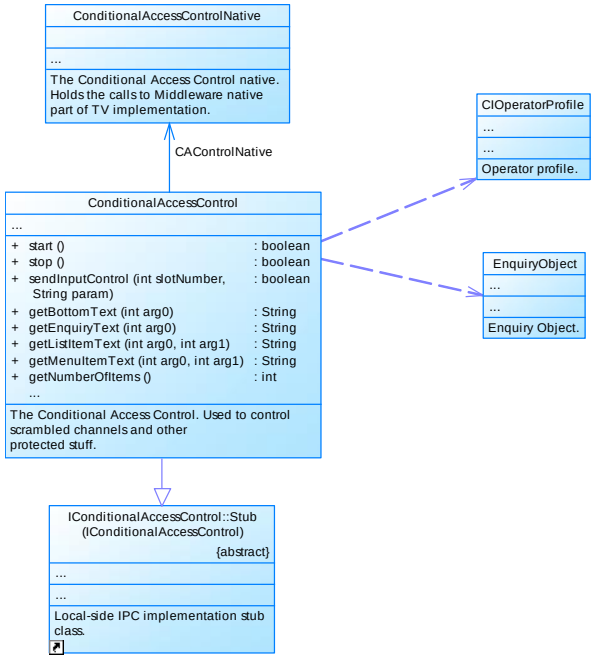
Obezbeđuje klase neophodne za kontrolu audio podataka u TV aplikaciji.



Slika B.2 Sadržaj paketa android.dtv.audio

Paket: android.dtv.ca

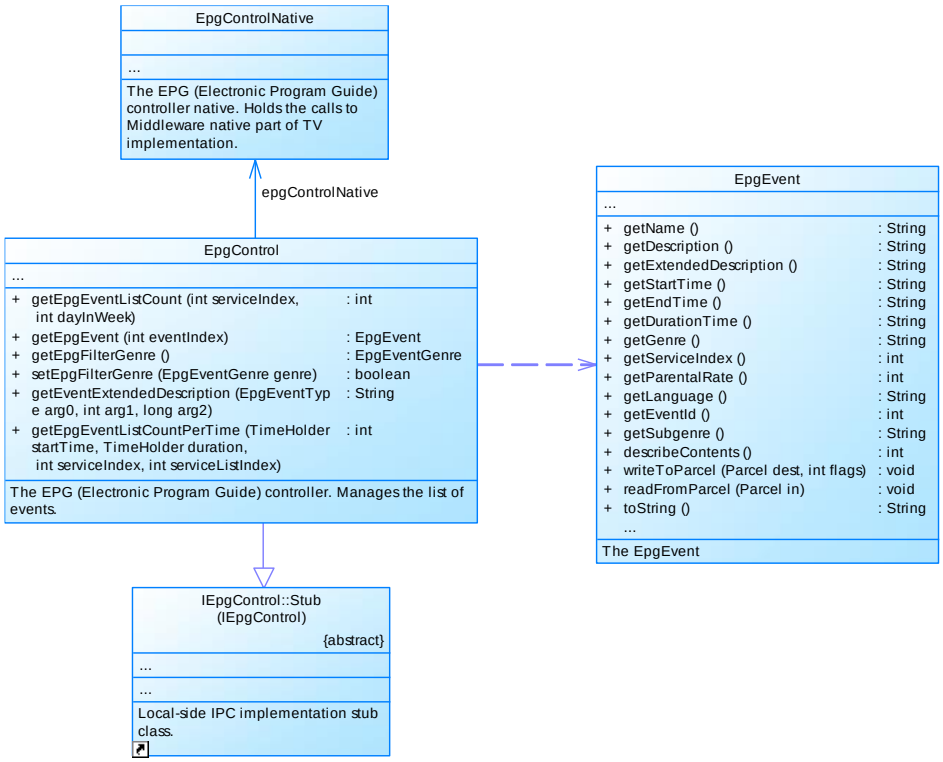
Obezbeđuje klase neophodne za upravljanje kontrolisanim pristupom (CA) u TV aplikaciji.



Slika B.3 Sadržaj paketa android.dtv.ca

Paket: android.dtv.epg

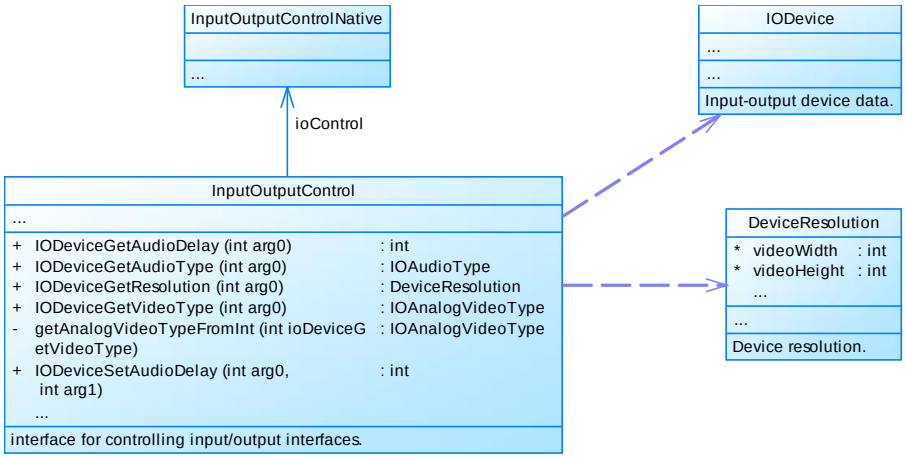
Obezbeđuje klase neophodne za upravljanje elektronskim programskim vodičem (EPG) u TV aplikaciji.



Slika B.4 Sadržaj paketa android.dtv.epg

Paket: android.dtv.io

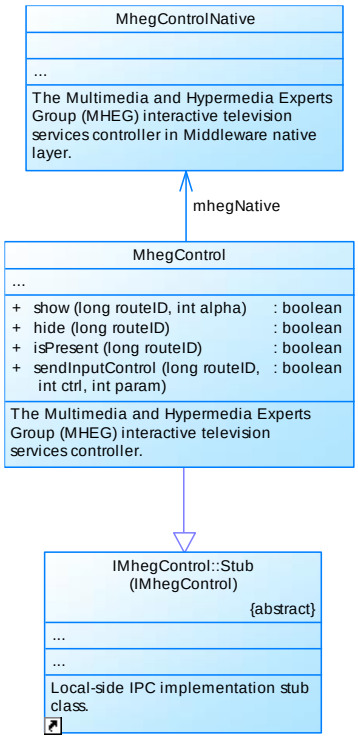
Obezbeđuje klase neophodne za kontrolu ulaznih/izlaznih sprega u TV aplikaciji.



Slika B.5 Sadržaj paketa android.dtv.io

Paket: android.dtv.mheg

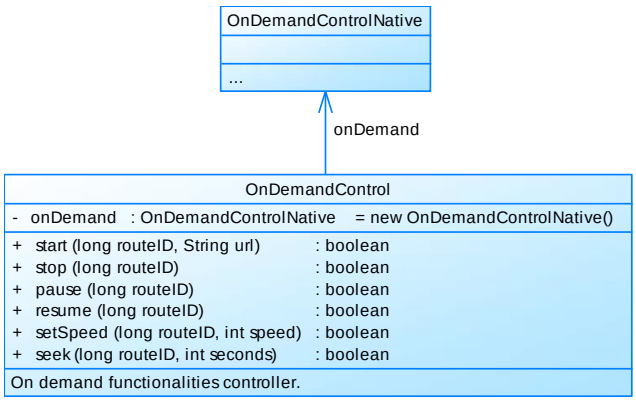
Obezbeđuje klase neophodne za upravljanje MHEG interaktivnim TV servisima u TV aplikaciji.



Slika B.6 Sadržaj paketa android.dtv.mheg

Paket: android.dtv.ondemand

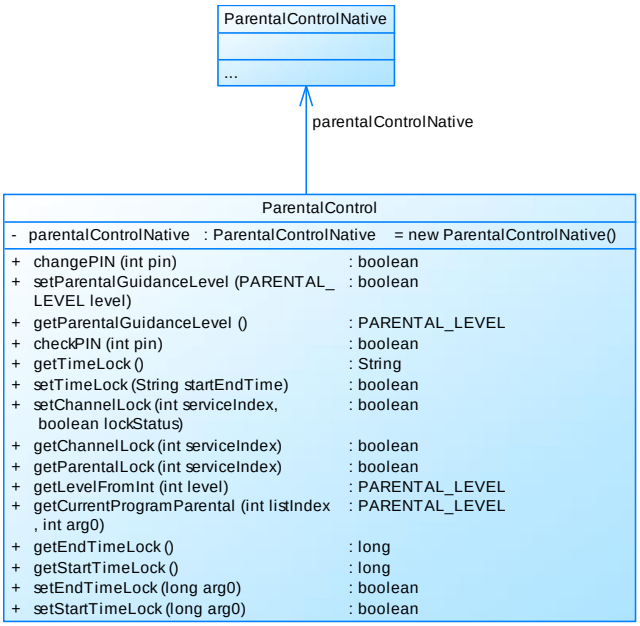
Obezbeđuje klase neophodne za upravljanje funcionalnošću video na zahtev u TV aplikaciji.



Slika B.7 Sadržaj paketa android.dtv.ondemand

Paket: android.dtv.parental

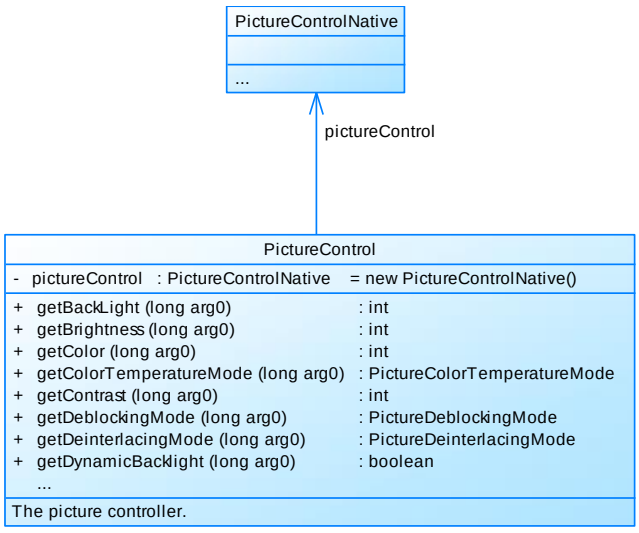
Obezbeđuje klase neophodne za upravljanje funcionalnošću roditeljske kontrole u TV aplikaciji.



Slika B.8 Sadržaj paketa android.dtv.parental

Paket: android.dtv.picture

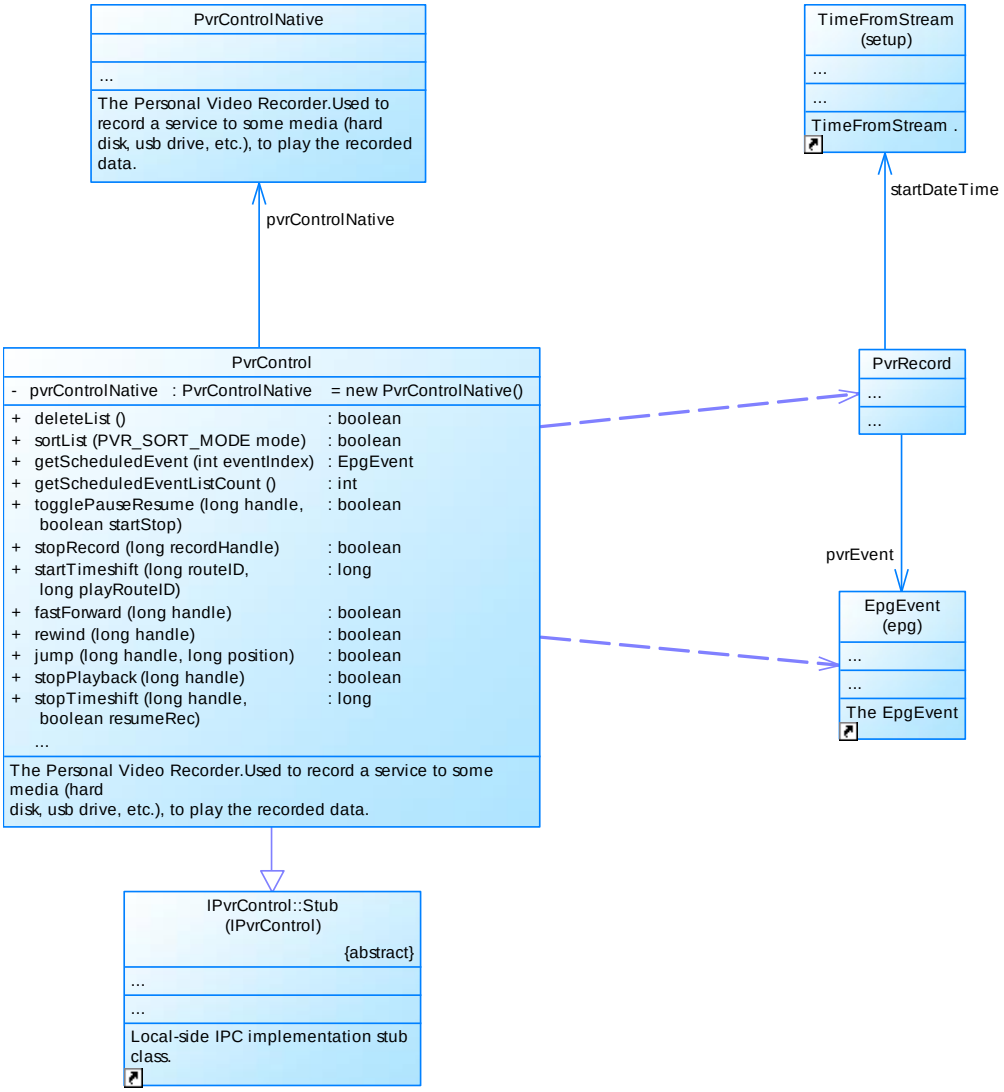
Obezbeđuje klase neophodne za upravljanje specijalnim podešavanjima izlazne slike u TV aplikaciji.



Slika B.9 Sadržaj paketa android.dtv.picture

Paket: android.dtv.pvr

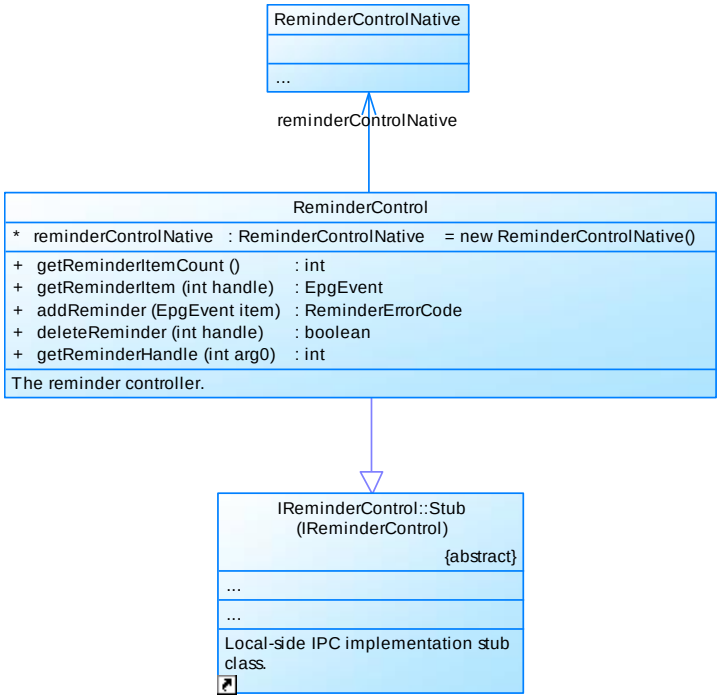
Obezbeđuje klase neophodne za upravljanjem personalnim video snimačem (PVR) u TV aplikaciji.



Slika B.10 Sadržaj paketa android.dtv.pvr

Paket: android.dtv.reminder

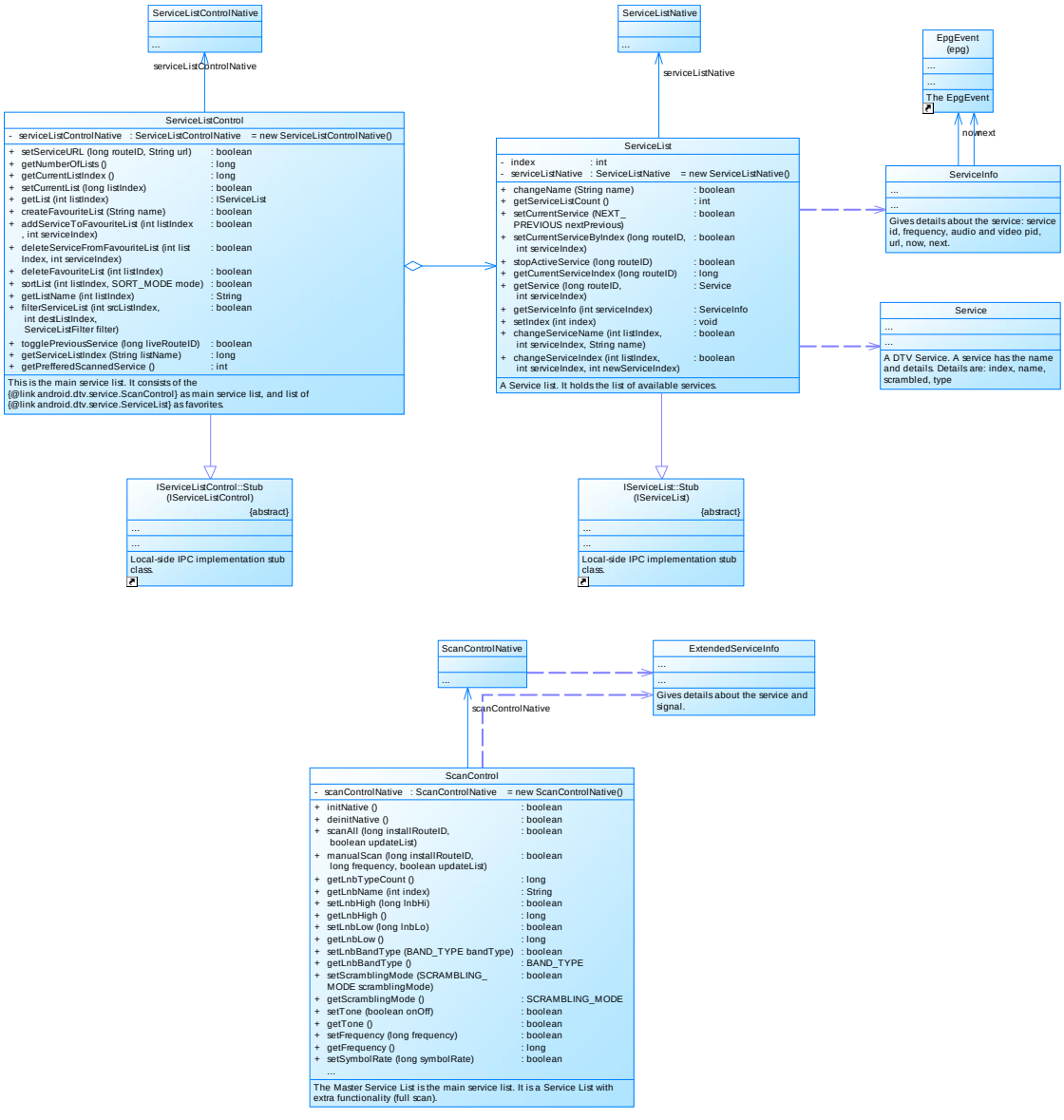
Obezbeđuje klase neophodne za upravljanje funkcionalnošću podsetnika u TV aplikaciji.



Slika B.11 Sadržaj paketa android.dtv.reminder

Paket: android.dtv.service

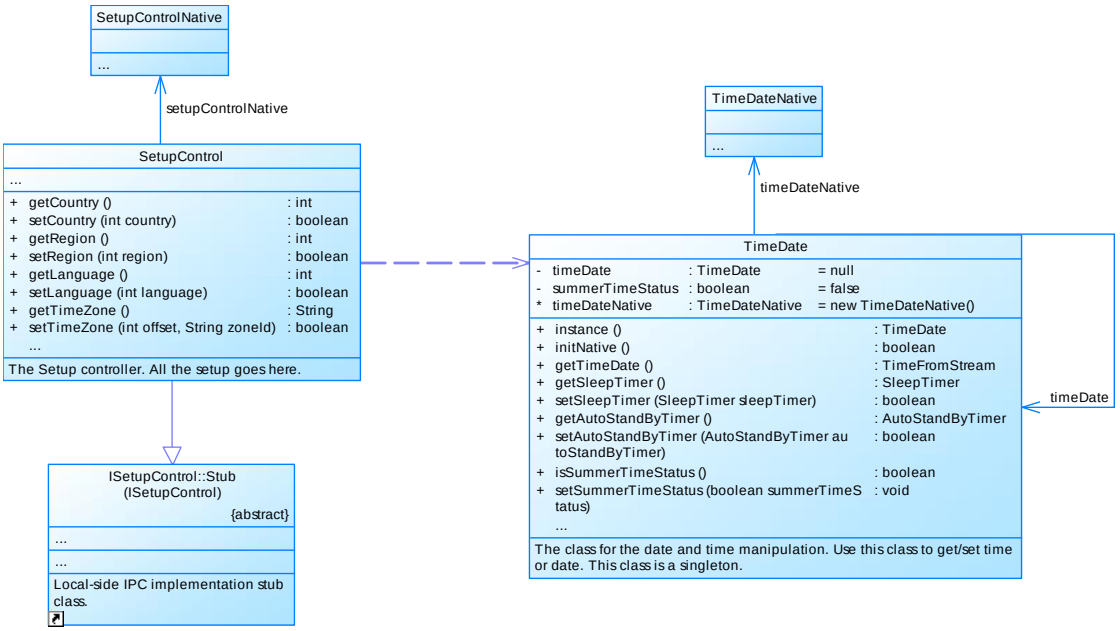
Obezbeđuje klase neophodne za upravljanje servisnim listama u TV aplikaciji.



Slika B.12 Sadržaj paketa android.dtv.service

Paket: android.dtv.setup

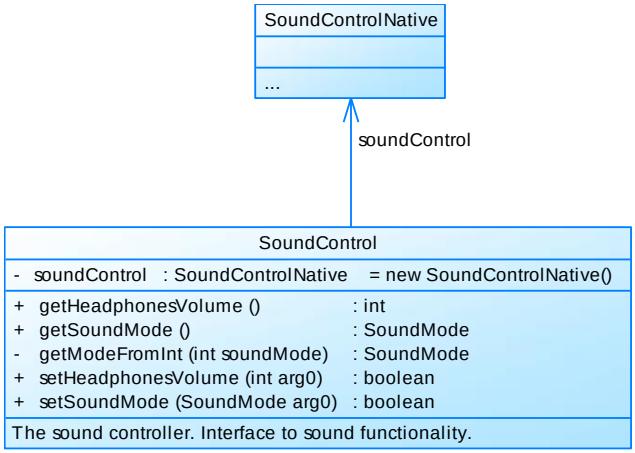
Obezbeđuje klase neophodne za konfigurisanje TV prijemnika iz TV aplikacije.



Slika B.13 Sadržaj paketa android.dtv.setup

Paket: android.dtv.sound

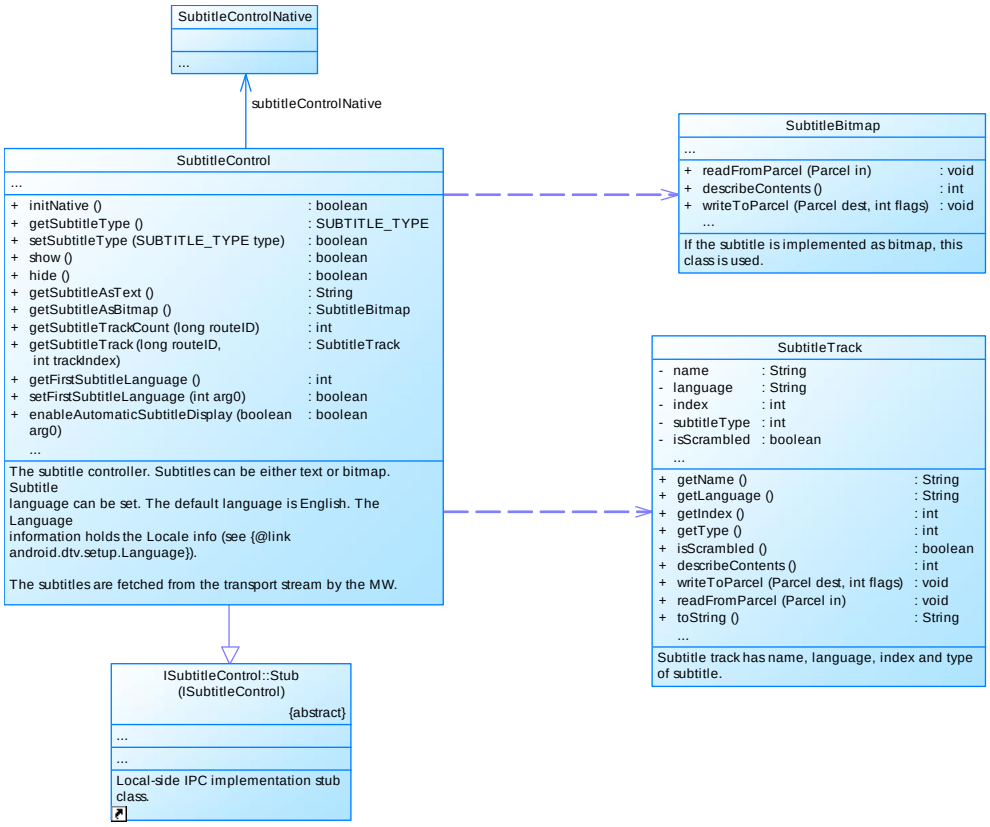
Obezbeđuje klase neophodne za upravljanje podešavanjima izlaznog zvuka TV platforme iz TV aplikacije.



Slika B.14 Sadržaj paketa android.dtv.sound

Paket: android.dtv.subtitle

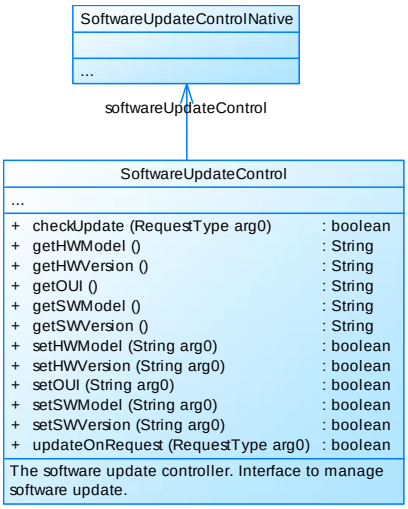
Obezbeđuje klase neophodne za upravljanje prevodima iz TV aplikacije.



Slika B.15 Sadržaj paketa android.dtv.subtitle

Paket: android.dtv.swupdate

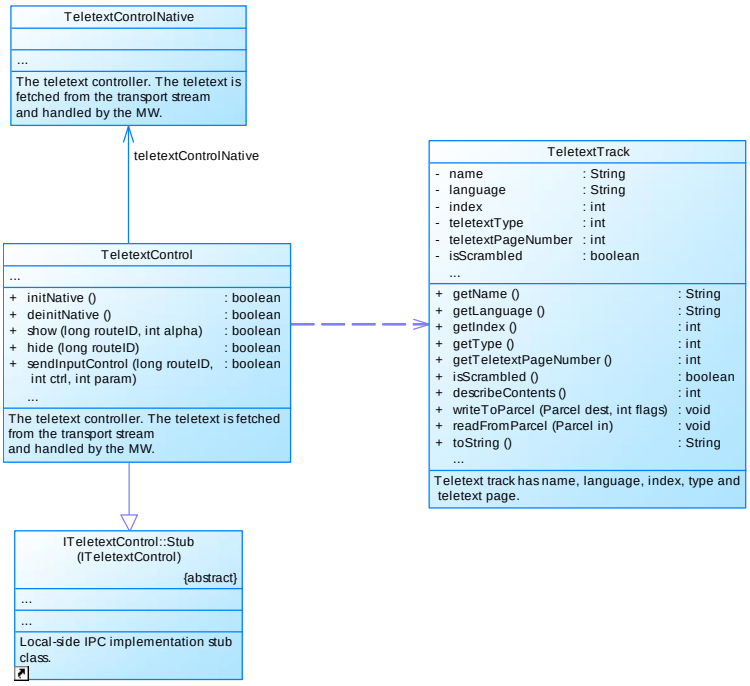
Obezbeđuje klase neophodne za upravljanje mehanizmom ažuriranja programske podrške TV prijemnika.



Slika B.16 Sadržaj paketa android.dtv.swupdate

Paket: android.dtv.teletext

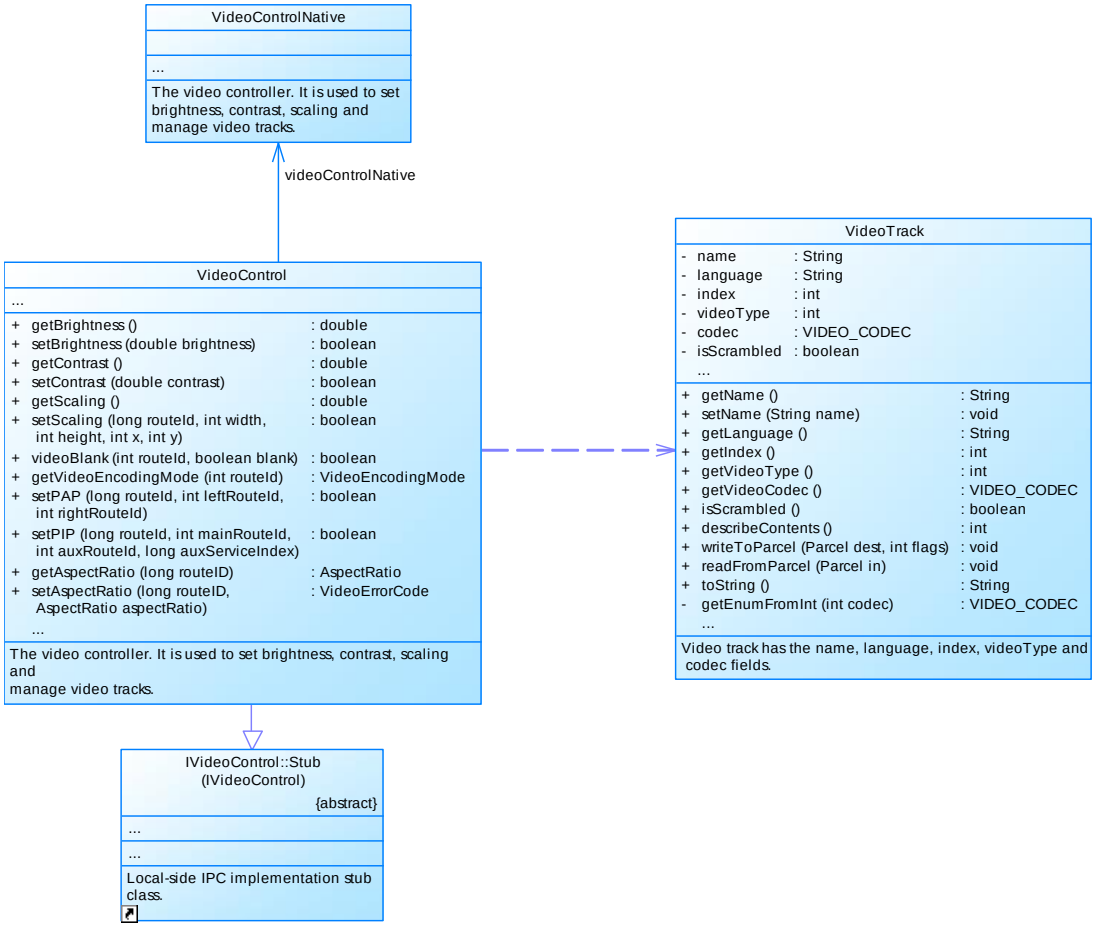
Obezbeđuje klase neophodne za upravljanje funkcionalnošću teleteksta iz TV aplikacije.



Slika B.17 Sadržaj paketa android.dtv.teletext

Paket: android.dtv.video

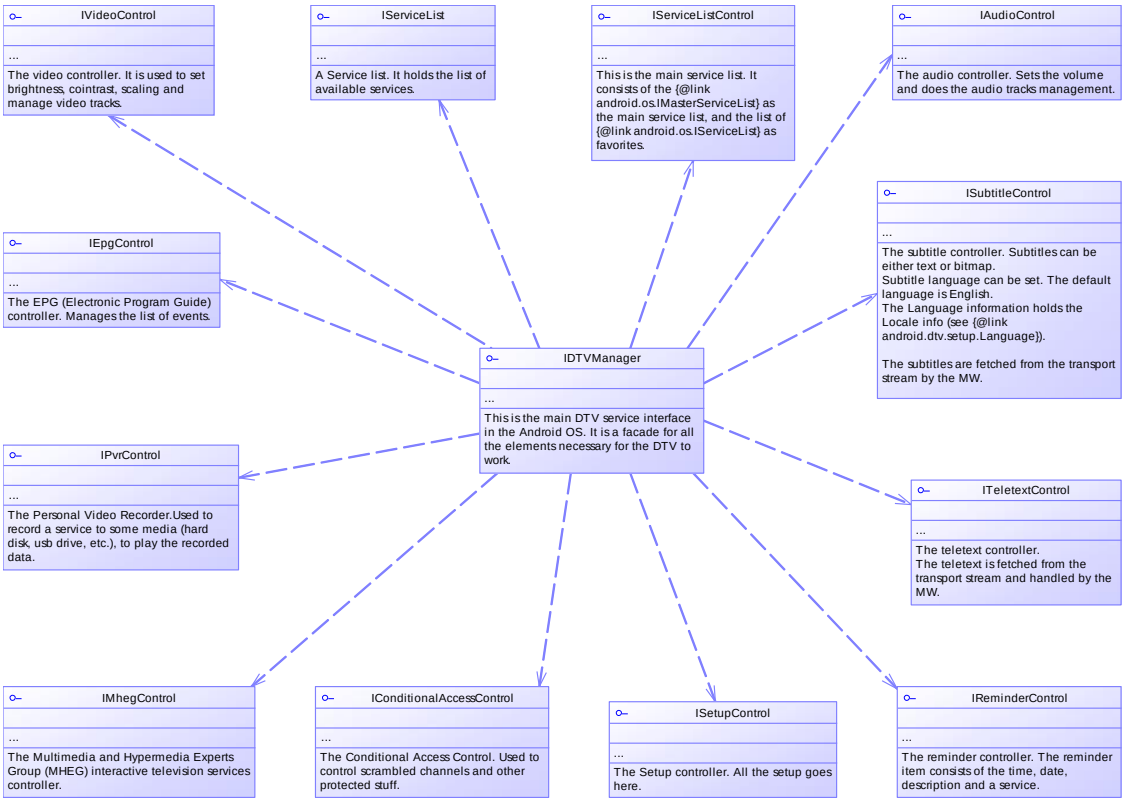
Obezbeđuje klase neophodne za upravljanje podešavanjima izlaznog video signala TV platforme iz TV aplikacije.



Slika B.18 Sadržaj paketa android.dtv.video

Paket: android.os

Obezbeđuje sprege neophodne za upravljanje TV aplikacijom.



Slika B.19 Sadržaj paketa android.os