UDC: 004.42:316.774 Info M: str. 28-34

TEHNIKE ZA RJEŠAVANJE PROBLEMA ANDROID FRAGMENTACIJE
TECHNIQUES FOR SOLVING OF ANDROID FRAGMENTATION PROBLEM

Nikola Obradović, prof. dr Zoran Đurić

REZIME: U ovom radu analiziran je razvoj Android aplikacija, kao i svi dominantni vidovi fragmentacije koji se pojavljuju prilikom njihovog razvoja. U radu je dat uopšten pogled na problematiku fragmentacije te osnovna definicija i podjele fragmentacije, analiza tipova fragmentacije Android platforme, te tehnika za rješavanje problema. Opisani su tipovi fragmentacije koji su najizraženiji, fragmentacija operativnog sistema i fragmentacija hardvera, te su analizirani osnovni mehanizmi defragmentacije. Na kraju predstavljen je i analizator koji praktično demonstrira da li su opisane tehnike primjenjene u nekim od najpoznatijih Android aplikacija otvorenog koda te je, na osnovu raspoloživih podataka, izvršena statistička analiza aplikacija koja prikazuje procentualni gubitak tržišta zbog problema fragmentacije.

KLJUČNE REČI: mobilni uređaji, Android, fragmentacija

ABSTRACT: In this paper, the development of Android applications, as well as all the dominant forms of fragmentation that can occur during application development have been analyzed. The paper gives a general view of the problem of fragmentation and basic definition and divisions of fragmentation, analysis of the types of fragmentation of the Android platform, and techniques for problem solving. There are described types of fragmentation that are the most featured, fragmentation of the operating system and hardware fragmentation, and than analyzed basic defragmentation mechanisms. Finally, it is presented analyzer that practically demonstrates whether the described techniques, based on the available data, are applied in some of the most popular open source Android applications, and then it is performed statistical analysis of applications that shows the market percentage loss due to the fragmentation.

KEY WORDS: mobile devices, Android, fragmentation

1. UVOD

Tržište mobilnih uređaja i mobilnih aplikacija doživljava najveću ekspanziju u svojoj istoriji. Razvojem ovog tržišta pojavio se veliki broj proizvođača, kako hardvera tako i softvera, koji učestvuju u razvoju, a sa željom da zauzmu dio tržišta. Svakodnevnim kreiranjem novih proizvoda raste i raznolikost opcija i mogućnosti mobilnih uređaja, pogotovo kada u razvoju istog ili sličnog proizvoda učestvuje veći broj proizvođača, kao što je slučaj sa Android platformom.

Android je Linux bazirani operativni sistem [1] za mobilne uređaje razvijen u saradnji kompanije Google i OHA (*eng. Open Handset Alliance*) konzorcijuma. Na osnovu istraživanja tržišta Android je u roku od nekoliko godina postao najkorišteniji operativni sistem na mobilnim uređajima [2]. Izvorni kod ovog operativnog sistema je javno dostupan, što znači da ga svako može modifikovati i prilagođavati uređajima koje proizvodi. Međutim, kreiranje ovakve otvorene platforme koja kao posljedicu ima poliferaciju uređaja od strane velikog broja proizvođača, dovelo je do problema fragmentacije.

Suštinski, fragmentacija označava različito ponašanje aplikacije na svakom uređaju na kojem se aplikacija koristi [3]. Bolje rečeno, fragmentacija je nemogućnost kreiranja aplikacije otporne na operativni kontekst i postizanja planiranog ponašanja na svim operativnim kontekstima za navedenu aplikaciju [4]. Pri tome operativni kontekst za aplikaciju predstavlja eksterno okruženje koje utiče na njeno funkcionisanje. Samim tim operativni kontekst je definisan hardverskim i softverskim okruženjem uređaja, korisnikom aplikacije kao i drugim faktorima (poput operatora mobilne telefonije).

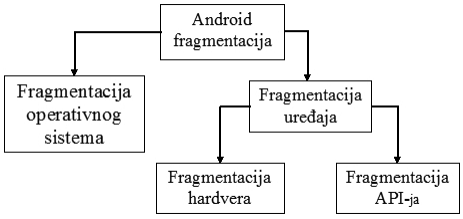
Pojedinci fragmentaciju definišu kao veliki broj verzija operativnog sistema, drugi kao opcionalni, dodatni, API (*eng. Application Programming Interface*) koji uzrokuje nekonzistentne implementacije platforme, a treći kao problem u razlikama hardvera i opcijama koje svaki pojedinačni uređaj nudi [3].

Očigledno je da fragmentacija utiče na gotovo sve aktere tržišta mobilnih uređaja. Tu su prvenstveno korisnici aplikacija, programeri, distributeri, telekom operateri i proizvođači uređaja. Tokom razvoja aplikacija fragmentacija može uticati na sve faze razvoja softvera poput upravljanja zahtjevima, analize i dizajna, implementacije, testiranja, upravljanja projektom, upravljanja konfiguracijama i verzijama, itd. U sekciji 2 opisani su tipovi Android fragmentacije, dok su u sekciji 3 dati osnovni mehanizmi Android defragmentacije. Fragmentacija operativnog sistema i mehanizmi defragementacije dati su u sekciji 4, a framentacija hardvera i odgovarajući mehanizmi defragmentacije dati su u sekciji 5. Prije zaključka, u sekciji 6, predstavljen je analizator koji je razvijen u toku istraživanja, a koji analizira Andorid aplikaciju i prikazuje broj uređaja sa kojima aplikacija neće biti kompatabilna ili na kojima korisničko iskustvo neće biti na zadovoljavajućem nivou, izražen u procentima.

2. OSNOVNA PODJELA ANDROID FRAGMENTACIJE

Kako je fragmentacija posljedica diverziteta najbolje je po osnovnim tipovima diverziteta tražiti i rješenje za koje aplikacija ima predviđen način rada u svim operativnim kontekstima. Posmatrajući tržište mobilnih uređaja očigledno je da postoje mnogi faktori, tačnije operativni konteksti, koji utiču na aplikacije koje se izvršavaju na mobilnom uređaju.

Prepoznata kao i podjela kod drugih [3], osnovna podjela Android fragmentacije je na fragmentaciju operativnog sistema i fragmentaciju uređaja. Fragmentacija uređaja se dalje dijeli na fragmentaciju hardvera i fragmentaciju API-ija. Šema podjele je data na slici 1.



Slika 1: Osnovna podjela Android fragmentacije [3]

Fragmentacija operativnog sistema predstavlja fragmentaciju usljed različitih verzija operativnog sistema. Manifestuje se kroz problem kompatabilnosti aplikacije sa Android operativnim sistemom koje su posljedica čestog izdavanja novih verzija operativnog sistema.

U sklopu fragmentacije uređaja fragmentacija hardvera podrazumijeva da među velikim brojem uređaja postoje različite hardverske specifikacije. Fragmentacija API-ja je problem prilikom kojeg svaki proizvođač modifikuje API za potrebe svoga uređaja.

Razvoj Android aplikacija koje su postigle potpunu ili gotovo potpunu eliminaciju problema fragmentacije postaje sve teži i skuplji, a biće još teži imajući u vidu proizvodnju sve većeg broja uređaja, direktno usmjerenih kupcu, različitih tipova i različitih verzija Android operativnih sistema.

3. OSNOVNI MEHANIZMI ANDROID DEFRAGMENTACIJE

Postoje tri ključne komponente koje omogućavaju osnovnu defragmentaciju. Prva komponenta je program kompatabilnosti Androida (*eng. Android Compatability Program*) uz pomoć kojeg se definišu tehnički detalji platforme i kreiraju alati koje koriste OEM (*eng. Original Equipment Manufacturer*) proizvođači kako bi se osiguralo da aplikacije funkcionišu na različitim uređajima. Druga komponenta je Android SDK koji posjeduje ugrađene alate koje koriste programeri kako bi definisali koje funkcionalnosti uređaja zahtjeva aplikacija koju prave. Treća komponenta je Google Play koji prikazuje krajnjim korisnicima samo one aplikacije koje se mogu pravilno izvršavati na njihovim uređajima [5].

Filtriranje na Google Play-u[1] je bazirano na određenom broju metapodataka aplikcije, ali i konfiguracionih podešavanja, poput deklaracija u manifestu, obaveznih biblioteka, arhitekturalnih zavisnosti i distribucionog seta u Google Play konzoli za programere (*eng. Google Play Developers Console*)[2], poput geografskog targetiranja [6]. Prilikom odluke da li će se određena aplikacija prikazati na nekome uređaju Google Play provjerava hardverske i softverske zahtjeve aplikacije, ali i telekom operatera, lokaciju i druge karakteristike. Ukoliko je aplikacija kompatabilna sa uređajem, a na osnovu pravila u filteru, aplikacija će biti dostupna. U suprotnom slučaju aplikacija neće biti dostupna na Google Play-u za navedeni uređaj.

Većina filtrtiranja se vrši na osnovu AndroidManifest.xml fajla, gdje se može izvršiti filtriranje uz pomoć odgovarajućih manifest elemenata [6]. Nakon publikovanja aplikacije Google Play na osnovu atributa unutar manifest elemenata određuje kojim uređajima će aplikacija biti dostupna, a kojima neće. Manifest elementi koji se tom prilikom koriste su: `<supports-screens>`, `<uses-configuration>`, `<uses-feature>`, `<uses-library>`, `<uses-permission>`, `<uses-sdk>`, `<compatible-screens>` i `<supports-gl-texture>`.

4. FRAGMENTACIJA OPERATIVNOG SISTEMA I MEHANIZMI DEFRAGMENTACIJE

Kada je u pitanju fragmentacija operativnog sistema postoje analize [7] koje se tiču distribucije verzija operativnih sistema kroz mjesece i frekventnosti njihovih instalacija. Google periodično generiše izvještaje o broju instalacija što su mnogi iskoristili kao podlogu za pravljenje analiza vezanih za problem fragmentacije. Analize su bile zasnovane na Kurtosis-u[3] i Herfindahl-Hirschman-ovom indeksu[4]. Analizom je pokazano da je problem fragmentacije Android operativnog sistema najizraženiji ubrzo nakon izdavanja nove verzije operativnog sistema, te da dostiže minimum malo prije izdavanja nove verzije operativnog sistema.

Takođe, pokazano je da fragmentacija Android operativnog sistema poprima ciklični karakter, te da ovi ciklusi dobijaju svoju godišnju učestalost, što je u skladu sa politikom Google-a da se objavljuje jedna nova verzija operativnog sistema na godišnjem nivou. Ispostavilo se da ovi ciklusi omogućavaju programerima da prilagode aplikaciju za novu verziju operativnog sistema na vrijeme, jer je potrebno oko 6 mjeseci da nova verzija počne zauzimati ulogu na tržištu nakon objavljivanja [7].

Pokazano je i da se prilikom održavanja i razvoja aplikacija može fokusirati na samo dvije najnovije verzije Android operativnog sistema (misli se na dvije verzije operativnog sistema prije verzije koja se tek pojavila na tržištu ili se najavljuje), te kreirati aplikacije za iste i na taj način uveliko ublažiti problem fragmentacije. Nakon toga potrebno je samo fokusirati se na podršku za najnoviju verziju koja će zauzimati sve veći udio na tržištu.

[1] Postoje i drugi marketi za distribuciju Android aplikacija ali Google Play trenutno zauzima primat a i drugi marketi u velikom obimu koriste isti ili sličan način filtriranja

[2] Google Play konzola za programere predstavlja centralno mjesto za distribuciju aplikacija, kreiranih od strane programera, sa setom mnogih podešavanja koji omogućavaju kvalitetno prezentovanje aplikacije na tržištu.

[3] Kurtosis predstavlja relativnu zašiljenost ili spljoštenost raspodjele u odnosu na normalnu raspodjelu.

[4] Herfindahl-Hirschman-ov indeks se koristi za mjerenje koncentracije entiteta na tržištu, sa ciljem mjerenja konkurentnosti, odnosno analize monopola na tržištu.

Očigledno je da bi bilo dobro, pored detaljne analize, dati i prioritete u pogledu defragmentacije verzija operativnog sistema. Ako, na primjer, samo nekoliko verzija operativnog sistema zauzima veliki procenat tržišta, to znači da bi se njima trebao dati prioritet. Trenutno vodeće verzije (Lollipop, KitKat, Jelly Bean, Ice Cream Sandwich, Honeycomb, Gingerbread) Android operativnog sistema zauzimaju oko 95% tržišta [8]. Pri tome osjetan je konstantan pad učešća starijih verzija u odnosu na novije verzije.

Android aplikacije su uglavnom kompatabilne prema novijim verzijama operativnog sistema. Međutim, aplikacije mogu, ali ne moraju biti kompatabilne sa starijim verzijama operativnog sistema. Razlog leži u tome što nove verzije operativnog sistema dodaju funkcionalnosti koje ne postoje u starijim verzijama operativnog sistema. Svaka nova verzija Android operativnog sistema sa sobom donosi i novu verziju API-ja koju programeri mogu da koriste, a samim tim i funkcionalnosti koje prije toga nisu ni postojale. Sa druge strane proizvođači mobilnih telefona vrlo često ne izdaju nove verzije Android operativnog sistema za svoje uređaje, što kao posljedicu dovodi do problema da nove funkcionalnosti u API-iju nisu ni podržane od strane takvog uređaja.

Stoga je Google morao da ponudi rješenja za ovaj problem. Jedno rješenje su Google Play Servisi (*eng. Google Play Services*) [9]. Iako je u pitanju Android aplikacija, ona je znatno više od toga. Google Play Servisi su zapravo nova Googleova platforma. Od objavljivanja prve verzije aplikacije Google Play Servisa Google sa svakom novom verzijom prebacuje određene funkcionalnosti, dijelove API-ija i druge važne elemente iz Android operativnog sistema u prethodno spomenutu aplikaciju. Aplikacija Google Play Servisa sadrži svoju verziju Google servisa i radi kao pozadinski servis Anrdoid operativnog sistema. Svim uređajima od verzije Androida Gingerbread je uz pomoć ove aplikacije omogućeno da redovno primaju nova ažuriranja za određene funkcionalnosti. Na ovaj način Google Play Servisi programeru daju slobodu da koriste najnovije API-je za popularne Google servise bez brige o mnogim problemima vezanim za fragmentaciju. Međutim, ovakvo rješenje ima i potencijalne mane. Najveća se ogleda u tome da Google Play Servisi nisu otvorenog koda već su u vlasništvu Google-a. To implicira da je Android operativni sistem sve manje operativni sistem otvorenog koda.

Najstariji operativni sistem koji ima veći procentualni udio u tržištu je Gingerbread (API 10), a to znači da je potrebno kreirati aplikacije koje se mogu izvršavati i na verziji API-ja 10. Međutim, kao što smo već spomenuli, kod takvih aplikacija funkcionalnosti koje su došle sa novim verzijama API-ja neće biti dostupne. Najvažnije funkcionalnosti na koje se neće moći računati u verziji Gingerbread su loaderi (*eng. Loaders*), fragmenti (*eng. Fragments*) i akcioni bar (*eng. ActionBar*) [10].

Pored Google Play Servisa, za rješavanje ove problematike, ali i drugih nekompatabilnosti u funkcionalnostima između verzija koriste se biblioteke podrške. Među njima prednjači zvanična biblioteka podrške (*eng. Android support library*) [11], kreirana od strane proizvođača Android operativnog sistema. Zvanična biblioteka podrške se izdaje u različitim verzijama. Tako, na primjer, verzija v4 predstavlja podršku unazad do verzije API-ja 4, a verzija v13 predstavlja podršku unazad do verzije API-ja 13. Pri tome je važno napomenuti da v13 sadrži sve što sadrži i v4, ali i dodatne klase za rad sa API-jima jednakim ili višim od verzije 13. Najveća primjena, pogotovo v4 biblioteke, odnosi se na izmjene koje su došle sa verzijom Androida 3.0 Honeycomb (API 11). Najvažnije klase su: `Fragment`, `FragmentManager`, `FragmentTransaction`, `List Fragment`, `DialogFragment`, `LoaderManager`, `Loader`, `AsyncTaskLoader` i `CursorLoader`. Važno je napomenuti da se redovno izdaju i druge verzije zvanične biblioteke podrške shodno operativnim kontektstima za koje je ista potrebna (npr. v17 Leanback biblioteka je kreirana kao podrška za razvoj korisničkih interfejsa aplikacija koje bi bile korištene na televizorima).

Zvanična biblioteka podrške ne podržava sve najnovije funkcionalnosti koje je potrebno defragmentirati, te se redovno ažurira sa ciljem kreiranja podrške i za najnovije funkcionalnosti. Kao posljedicu toga imamo da se u međuvremenu za određene neimplementirane funkcionalnosti kreiraju i koriste druge alternativne biblioteke podrške. Tipičan primjer je podrška za akcioni bar koja nije postojala u starijim verzijama zvanične biblioteke podrške. Stoga je kreirana biblioteka *ActionBarSherlock* koja predstavlja ekstenziju za zvaničnu biblioteku podrške, dizajnirana za upotrebu akcionog bara u svim verzijama Androida od verzije API-ja 5 pa na više [12]. U međuvremenu je i u sklopu zvanične biblioteke podrške implementirana ova funkcionalnost.

5. FRAGMENTACIJA HARDVERA I MEHANIZMI DEFRAGMENTACIJE

Problem fragmentacije hardvera proizilazi iz razlika samih uređaja koji posjeduju ekrane sa različitim karakteristikama, različite arhitekture procesora, različitu namjenu uređaja koji koristi Android operativni sistem, ali i različite hardverske funkcionalnosti (npr. neki uređaji posjeduju kameru, a neki ne).

Najizraženiji problem je fragmentacija vezana za korisnički interfejs, koja je direktna posljedica različitih karakteristika ekrana uređaja i o čemu se govori u nastavku. U ovom slučaju Android je obezbjedio mehanizme koji omogućavaju kreiranje relativno konzistentnog korisničkog interfejsa aplikacije za sve uređaje. Android posjeduje API koji obezbjeđuje kontrolu korisničkog interfejsa aplikacije, a sve sa ciljem optimizacije korisničkog interfejsa za različite konfiguracije ekrana.

Ukoliko se i ne primjenjuju mehanizmi obezbjeđeni Android API-ijem sistem će samostalno izvršiti skaliranje korisničkog interfejsa aplikacije i prikazati isti korisniku. Međutim, na ovaj način korisnik neće imati osjećaj da je aplikacija kreirana za njega te će korisničko iskustvo u upotrebi aplikacije biti lošije.

Pošto su kao problem prepoznati i određeni pojmovi koji su u vezi sa karakteristikama ekrana date su i definicije istih, a sve sa ciljem lakšeg razumijevanja mehanizama Adroid API-ja.

Veličina ekrana predstavlja fizičku veličinu, mjerenu po dijagonali ekrana. Zbog jednostavnosti Android grupiše sve veličine ekrana u četiri kategorije: mali, normalni, veliki i veoma veliki (*eng. small, normal, large i extra large, često nazivana i xlarge*).

Gustina ekrana (*eng. screen density*) predstavlja broj piksela u fizičkom području ekrana. Vrlo često se odnosi na dpi ili broj tačaka po inču (*eng. dots per inch*). Android grupiše sve gustine ekrana u šest kategorija: niska, srednja, visoka, veoma visoka, veoma-veoma visoka i veoma-veoma-veoma visoka (*eng. low ili ldpi, medium ili mdpi, high ili hdpi, extra high ili xhdpi, extra-extra high ili xxhdpi, extra-extra-extra high ili xxxhdpi*).

Orijentacija predstavlja orijentaciju ekrana sa korisnikove tačke gledišta. Ekran može biti orijentisan kao pejsaž (*eng. landscape*) ili portret (*eng. portrait*), a samim tim i korisnički interfejs aplikacije. Bitno je napomenuti da uređaji nemaju istu podrazumjevanu orijentaciju, kao i da mnogi uređaji imaju mogućnost promjene orijentacije.

Rezolucija se definiše kao ukupan broj piksela na ekranu. Prilikom dodavanja podrške za različite ekrane korisnički interfejs aplikacije ne vrši intrerakciju direktno sa rezolucijom, već sa veličinom i gustinom ekrana, kao i kategorijama istih.

Piksel nezavisan od gustine (*eng. density-independent pixel*) predstavlja virtuelnu jedinicu veličine koja se treba koristiti prilikom definisanja korisničkog interfejsa aplikacije, a sa ciljem definisanja dimenzija korisničkog interfejsa ili pozicije na način nezavisan od gustine. Kao jedinica mjere se koristi `dp`. Ovaj virtuelni piksel predstavlja ekvivalent jednom fizičkom pikselu na ekranu sa 160 tačaka (piksela) po inču (`dpi`). Poslije toga vrši se skaliranje `dp` jedinica čija je osnova trenutna gustina ekrana na kojoj se koristi aplikacija. U ovu svrhu primjenjuje se sljedeća formula:

```
px = dp * (dpi / 160)
```

Preporuka je da se uvijek koriste `dp` jedinice prilikom definisanja korisničkog interfejsa aplikacije jer će se na taj način korisnički interfejs na različitim ekranima prikazati pravilno [13].

Uz pomoć podjela na četiri kategorije veličine i šest kategorija gustine ekrana, može se reći da postoji težnja da se većina ekrana Android uređaja grupiše oko kategorije normalne veličine i srednje gustine ekrana. Međutim, kako je proizvođačima dozvoljeno odstupanje u rezoluciji, veličini ekrana i broju piksela po inču, primjetna je i fragmentacija.

Jedan od glavnih ciljeva prilikom izrade aplikacije je kreiranje korisničkog interfejsa koji nezavisan od gustine ekrana. Korisnički interfejs aplikacije je nezavisan od gustine ekrana ukoliko iz korisnikove tačke gledišta elementi interfejsa zadržavaju svoju veličinu u odnosu na gustinu ekrana uređaja.

Moglo bi se pretpostaviti da je usljed velikog broja različitih veličina ekrana i rezolucija ovo jedan od većih problema korisničkog interfejsa. Međutim, uz pomoć Android API-ja sistem sam skalira piksele nezavisne od gustine (`dp`), ali i grafičke resurse, poput slika, u zavisnosti od gustine ekrana na kojem se prikazuje korisnički interfejs aplikacije. Stoga je preporuka da se korisnički interfejs aplikacije kreira uz pomoć piksela nezavisnih od gustine (`dp`) i `wrap_content`-a[5] kada je god to moguće. Manji nedostatak u tome slučaju predstavlja skaliranje slika koje mogu biti iz toga razloga mutne ili pikselizovane, ali se isto nadopunjuje sa tim da se dostavljaju alternativni resursi za specifične slučajeve.

Kako je i ranije spomenuto, Android operativni sistem obavlja većinu aktivnosti oko prikaza korisničkog interfejsa aplikacije na najbolji mogući način, pod uslovom da se poštuju određena pravila. Pored prethodno pomenutog kreiranja korisničkog interfejsa uz pomoć piksela nezavisnog od gustine, tu uključujemo i sljedeće:

1. Eksplicitno definisati u manifestu koje veličine ekrana aplikacija podržava i na taj način vršiti filtriranje aplikacije. Isto se vrši uz pomoć manifest elementa `<supports-screens>`.
2. Definisati različite rasporede za različite veličine ekrana. Na ovaj način se korisnički interfejs prilagođava uređaju što vjerodostojnije.
3. Napraviti odvojene grafičke resurse u zavisnosti od veličine ekrana. U slučaju da se definišu grafički resursi samo za normalnu veličinu ekrana oni će biti skalirani za veće ili manje veličine ekrana. Na ovaj način stvara se i potencijalno loš korisnički interfejs aplikacije, koji utiče i na kvalitet iste.
4. Ne koristiti apsolutni raspored (`AbsoluteLayout`) jer je zastario i obavezuje korišćenje fiksiranih pozicija.

Android razvojno okruženje programerima omogućava da u aplikaciju integrišu alternativne resurse i na taj način u velikom broju slučajeva riješe problem fragmentacije, kao i posljedice koje nastaju usljed istog [14]. Prvenstveno se misli na rješenje da grafički resursi ne budu mutni ili pikselizovani, ali alternativni resursi mogu biti i za druge potrebe, poput jezika i slično.

Prilikom izvršavanja, Android operativni sistem koristi odgovarajući resurs zasnovan na trenutnoj konfiguraciji uređaja. Unutar projekta aplikacije potrebno je kreirati direktorijum `res/` u kojem se nakon toga kreiraju poddirektorijumi u koji će se spremiti alternativni resursi namjenjeni za određeni operativni kontekst. Sintaksa je sljedeća:

```
<naziv resursa>-<kvalifikator>
```

Pri tome `<naziv resursa>` je standardni naziv resursa koji se već nalazi u projektu aplikacije, poput `layout`, `drawable` i slično, a `<kvalifikator>` predstavlja konfiguracioni kvalifikator koji definiše specifičnu konfiguraciju uređaja. Na primjer, za slučaj tableta i potrebe za posebnim grafičkim resursima zbog veličine ekrana, pored `drawable` direktorijuma će se nalaziti i `drawable-hdpi` direktorijum u koji će se pohraniti resursi za ovakav i slične tipove uređaja. Pri tome `hdpi` znači da su ti resursi za uređaje visoke gustine ekrana.

Sa svakom novom verzijom operativnog sistema Google pokušava da umanji probleme fragmentacije. U najnovijoj

[5] `wrap_content` predstavlja jedan od parametara klase ViewGroup. LayoutParams i kao takav govori Androidu da prikaže pogled dovoljno velik tako da se prikaže sadržaj istoga. Na primjer, ukoliko govorimo o vidžetu (dugmetu) Button to bi značilo da je dugme (Button) toliko veliko taman da prikaže definisani tekst u istome.

verziji (Lollipop) jedan od najvećih doprinosa u borbi protiv fragmentacije je uvođenje materijalnog dizajna (*eng. Material design*) korisničkog interfejsa[15]. Cilj ovog dizajna je da omogući korisnicima konzistentan izgled svih aplikacija koje koriste, sa jedinstvenim korisničkim iskustvom bez obzira o kojoj platformi ili veličini ekrana je riječ. Ovo ne doprinosi samo boljem korisničkom iskustvu, već promoviše i adaptaciju postojećih aplikacija kako bi se smanjila kriva učenja za korištenje istih. Programeri će na ovaj način biti primorani da razmotre primjenu ovog novog modela korisničkog interfejsa u svojim aplikacijama. U suprotnom, ukoliko ne pređu na novi materijalni dizajn, aplikacija će izgledati zastarjelo. Kao direktnu posljedicu konkurentno okruženje većine Android marketa će primijetiti da aplikacija gubi na popularnosti, pogotovo kako se korisnici budu vremenom navikavali na novi dizajn.

6. OPIS RADA ANALIZATORA I REZULTATI SPROVEDENIH TESTOVA

Analizator koji predlažemo zasnovan je na opažanjima i preporukama spomenutim u prethodnim poglavljima. Cilj analizatora je da na osnovu relevantnih preporuka, a po pitanju fragmentacije, analizira Andorid aplikaciju i statistički prikaže procenat gubitka tržišta, odnosno procentualno izražen broj uređaja sa kojima aplikacija neće biti kompatabilna ili kod kojih korisničko iskustvo neće biti na zadovoljavajućem nivou. Značaj ovog rješenja ogleda se u tome što programerima daje uvid u fragmentaciju analizirane aplikacije i upozorava ih na mogući procentualni gubitak tržišta, te im daje preporuke za otklanjanje detektovanih problema.

Analizator preuzima sve podatke iz `AndroidManifest.xml` fajla koji mogu uticati na filtriranje aplikacije na Google Play-u ili loše korisničko iskustvo prilikom korišćenja aplikacije. Pri tome se koristi Java XML parser. Prvo se uzima podatak o nazivu paketa, a nakon toga se parsiraju sljedeći manifest elementi: `uses-sdk` - uzimaju se svi atributi koji postoje, `supports-screens` - uzimaju se svi atributi osim `android:requiresSmallestWidthDp`, `android:compatibleWidthLimitDp` i `android:largestWidthLimitDp`, `uses-configuration` - uzimaju se svi atributi, `uses-feature` – analizira se da li postoji taj element, `uses-permission` - analizira se da li postoji taj element, `compatible-screens` - uzimaju se svi atributi i kreira se matrica kompatabilnih ekrana i `supports-gl-texture` - analizira se da li postoji taj element. Parsiranje `AndroidManifest.xml` dokumenta i dodjeljivanje vrijednosti promjenljivim koje predstavljaju prethodno spomenute manifest elemente se vrši u klasi `XMLParser`.

Klasa `PercentageData` posjeduje sve promjenljive koje predstavljaju procentualne vrijednosti pojedinih veličina i gustina ekrana, verzija API-ja i verzija OpenGL-ES-a. Statistike su preuzete sa zvanične Google Play stranice zadužene za prikaz procentualnih udjela uređaja koji dijele određenu karakteristiku[6][8]. To znači da određene veličine ekrana sa određenom gustinom ili određena verzija API-ja neće biti u listi jer za iste ne postoje raspoloživi podaci. Za vrijednosti veličina ekrana su preuzete procentualne vrijednosti za male, normalne, velike i veoma velike. Za gustine ekrana su preuzete procentualne vrijednosti za nisku, srednju, visoku, veoma visoku, veoma veoma visoku i tv[7]. Takođe su i preuzete procentualne vrijednosti za postojeće veličine ekrana sa određenom gustinom. Procentualne vrijednosti su preuzete za sljedeće verzije API-ja: 10 (Gingerbread), 13 (Honeycomb) , 15 (Ice Cream Sandwich), 16 (Jelly Bean), 17 (Jelly Bean), 18 (Jelly Bean), 19 (KitKat) i (Lollipop) 21. Procentualne vrijednosti za OpenGL-ES su preuzete za verzije 1.1, 2.0 sa podrškom za 1.1 i 3.0 sa podrškom za 2.0 i 1.1.

Analizator vrši i analizu alternativnih resursa, pri čemu provjerava da li postoje alternativni direktorijumi vezani za grafiku: `drawable_hdpi`, `drawable_ldpi`, `drawable_mdpi`, `drawable_xhdpi`, `drawable_xxhdpi`, `drawable_tvdpi`, `values_sw600dp`, `values_sw720dp_land`. Pored toga provjerava se da li postoje i biblioteke podrške, `android-support-v4.jar` i `android-support-v13.jar`. Analizator u obzir uzima i sve jedinstvene identifikatore grafičkih resursa i postavlja u posebno kreiranu listu za taj slučaj, te vrši pregled svih klasa sa ciljem da ustanovi da li postoje klase koje su u vezi sa funkcionalnostima kod kojih je najviše izražen problem fragmentacije, kao što su loaderi i fragmenti.

Na osnovu prikupljenog podatka o minimalnoj i maksimalnoj zatjevanoj verziji API-ja računa se procentualni iznos onih verzija API-ja koje aplikacija ne podržava. U narednom koraku vrši se prikupljanje informacija o veličinama i gustinama ekrana koje aplikacija podržava. Za tu priliku koriste se manifest elementi `compatible-screens` i `supports-screens`. Prvi manifest element je dostupan od verzije API-ja 9, te je stoga potrebna provjera da li je isti i primjenljiv u aplikaciji. Ukoliko to jeste slučaj (element je prisutan i aplikacija podržava i starije verzije API-ja od verzije 9) pristupilo se tome da se kalkulacija vrši samo ako je verzija API-ja veća ili jednaka 9. Manifest element `compatable-screens` isključuje podršku za sve ostale ekrane osim onih koji se specifikuju, te samim tim isključuje i uticaj manifest elementa `supports-screens`. Manifest element `supports-screens` je predstavljen od verzije API-ja 4, verzije od koje se i vršila analiza. Na osnovu prethodno navedenog prikupljeni su podaci o tome koje veličine ekrana nisu podržane od strane aplikacije, te ukoliko postoje takvi slučajevi izvršena je analiza procenta gubitka tržišta. U slučaju kada je manifest element `compatable-screens` dostupan u `AndroidManifest.xml` fajlu, manifest element `supports-screens` se smatra nevažećim. Na sličan način se računa i procentualni gubitak tržišta zbog verzije OpenGL-a.

Potom se provjerava procenat onih uređaja na kojima korisničko iskustvo neće biti na zadovoljavajućem nivou (prvenstveno grafičko iskustvo). U ovom slučaju se ne govori o

[6] Uzete su samo procentualne vrijednosti koje su veće ili jednake 0.1%. Razlog zašto nisu uzimane procentualne vrijednosti manje od 0.1% je taj što ti podaci nisu bili dostupni.

[7] Gustine ekrana, veoma veoma visoka i tv, su se skoro pojavile i već postoje procentualne vrijednosti. Stoga su isto uključene u analizator, ali nisu spominjane u nekim prethodnim podjelama.

pravom gubitku korisnika i nemogućnosti instalacija aplikacija već o gubitku korisnika koji neće biti zadovoljni korisničkim iskustvom aplikacije što indirektno može uticati na loš rejting aplikacije. Prvo se vrši provjera postojanja datoteke za alternativne grafičke resurse. Za sve alternativne grafičke resurse se automatski računa procenat grafičkih resursa koji nisu adekvatni. Uzeta je u obzir pretpostavka da ukoliko postoje datoteke za alternativne resurse to ne mora značiti da iste posjeduju alternativne grafičke resurse. Iz tog razloga provjerava se da li sve datoteke sa alternativnim resursima posjeduju sve grafičke resurse koje aplikacija koristi i to sa ekstenzijama .png, .jpg i .gif. Listi grafičkih resursa koji su na raspolaganju pristupa se putem klase `R.java`[8]. Svi alternativni grafički resursi koji posjeduju manje od 80%[9] odgovarajućih resursa se smatraju neodgovarajućim, te se za iste uračunava procenat grafičkih resursa koji nisu adekvatni. Analizator, takođe, informiše koji alternativni resursi nisu pronađeni, a trebalo bi da postoje.

Nakon prethodnog vrši se i provjera da li aplikacija koristi neke od funkcionalnosti koje su dostupne u novijim verzijama API-ja, te koje nisu dostupne na starijim verzijama, ukoliko ne postoje biblioteke za podršku. Funkcionalnosti za koje je izvršena pretraga su fragmenti i loaderi. Analizator provjerava sve klase i druge resurse na način da nedvosmisleno dokazuje upotrebu prethodnih funkcionalnosti, te, ukoliko ne postoje biblioteke za podršku, upozorava na potencijalni problem, kao i mogući gubitak tržišta. Analizator nakon toga kreira i listu upozorenja u vezi sa filterima, ukoliko se koriste, i svim drugim prikupljenim podacima iz klasa jer isti takođe smanjuju potencijalno tržište aplikacije.

Na osnovu prikupljenih podataka, a u vezi sa dijelom tržišta koje bi se moglo izgubiti zbog nepodržavanja verzija API-ja, nepodržavanja određenih veličina i gustina ekrana, te OpenGL verzija, bilo je potrebno izračunati ukupan procentualni udio onih uređaja na kojima aplikacija neće biti dostupna. Ukoliko se prethodna tri problema posmatraju matematički kao tri skupa, očigledno je da su isti međusobno zavisni, te da postoje presjeci skupova koje čine zajednički slučajevi. Na primjer, postoji uređaj kojem aplikacija neće biti dostupna zbog nepodržavanja verzije API-ja, ali i zato što nije u listi podržanih veličina i gustina ekrana. Stoga je za kalkulaciju nad ta tri skupa korišćena formula uključenja-isključenja, koja je u opštem obliku predstavljena na sljedeći način:

$$\left|\bigcup_{i=1}^{n} A_i\right| = \sum_{i=1}^{n}|A_i| - \sum_{1\le i\le j\le n}\left|A_i \cap A_j\right| + \sum_{1\le i\le j\le k\le n}\left|A_i \cap A_j \cap A_k\right| - \ldots + (-1)^{n-1}\left|A_1 \cap \ldots \cap A_n\right|$$

pri čemu A_i predstavlja jedan od skupova od i do n. Za slučaj o kojem se govori u radu postoje tri skupa.

[8] R.java predstavlja dinamički generisanu klasu sa ciljem identifikovanja svih resursa u aplikaciji (grafički, stringovi, itd), kako bi se istima pristupilo iz drugih klasa unutar Android aplikacije.

[9] Vrijednost je odabrana lično i kao takva predstavlja procjenu do koje vrijednosti je tolerantno ne posjedovati određeni broj alternativnih resursa.

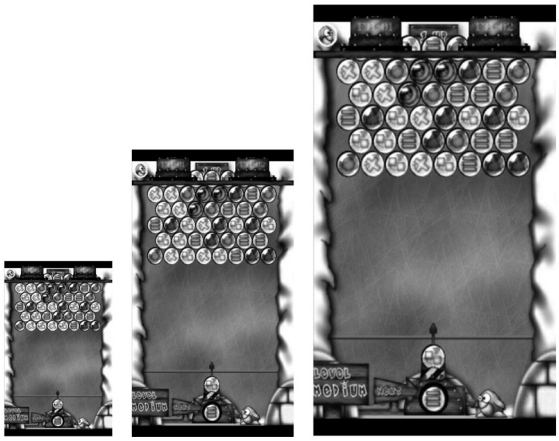
Kako se govori o procentu učešća pojedinih skupova u cjelini, što se može posmatrati i kao vjerovatnoća da se određeni događaj desi, procentualne vrijednosti prethodnih skupova su pretvorene u vrijednosti vjerovatnoće u rasponu od 0 do 1, a presjek dva skupa je predstavljen proizvodom vjerovatnoća tih skupova. Konačni rezultat je nakon toga ponovo pretvoren u procentualnu vrijednost. Mora se napomenuti da će ovaj način kalkulacije dati približan procenat gubitka tržišta za prethodne slučajeve, ali ne i tačan. Glavni razlog je postojanje mnogih drugih skupova, tipova fragmentacije, koji nisu ovom prilikom uključeni u analizu.

Za praktične primjere preuzeti su dostupni izvorni kodovi četiri popularne aplikacije, koje pripadaju različitim kategorijama. Aplikacije koje su analizirane su: SMS Backup+, Connectbot, HN-Android i Frozen Bubble. Analizator za svaku aplikaciju kreira i detaljan izvještaj u kojem su navedene sve fragmentacije u aplikaciji kao i procentualni gubici. U nastavku je dat dio izvještaja za aplikaciju SMS Backup+.

```
Getting all files in C:\src including those in
    subdirectories
file: C:\src\com\zegoggles\smssync\
    AccountManagerAuthActivity.java
file: C:\src\com\zegoggles\smssync\Alarms.java
...
file: C:\src\com\zegoggles\smssync\XOAuthConsumer.java
Resource is missing: /res/drawable-hdpi/ic_done
Resource is missing: /res/drawable-ldpi/ic_done
Resource is missing: /res/drawable-xhdpi/ic_done
Resource is missing: /res/drawable-xxhdpi/ic_done
Resource is missing: /res/drawable-xxxhdpi/ic_done
Resource is missing: /res/drawable-tvdpi/ic_done
Resource is missing: /res/drawable-hdpi/ic_error
Resource is missing: /res/drawable-ldpi/ic_error
Resource is missing: /res/drawable-xhdpi/ic_error
Resource is missing: /res/drawable-xxhdpi/ic_error
Resource is missing: /res/drawable-xxxhdpi/ic_error
Resource is missing: /res/drawable-tvdpi/ic_error
Resource is missing: /res/drawable-hdpi/ic_idle
Resource is missing: /res/drawable-ldpi/ic_idle
Resource is missing: /res/drawable-xhdpi/ic_idle
Resource is missing: /res/drawable-xxhdpi/ic_idle
Resource is missing: /res/drawable-xxxhdpi/ic_idle
Resource is missing: /res/drawable-tvdpi/ic_idle
Resource is missing: /res/drawable-ldpi/ic_launcher
Resource is missing: /res/drawable-xhdpi/ic_launcher
Resource is missing: /res/drawable-xxhdpi/ic_launcher
Resource is missing: /res/drawable-xxxhdpi/ic_launcher
Resource is missing: /res/drawable-tvdpi/ic_launcher
Resource is missing: /res/drawable-hdpi/ic_syncing
Resource is missing: /res/drawable-ldpi/ic_syncing
Resource is missing: /res/drawable-xhdpi/ic_syncing
Resource is missing: /res/drawable-xxhdpi/ic_syncing
Resource is missing: /res/drawable-xxxhdpi/ic_syncing
Resource is missing: /res/drawable-tvdpi/ic_syncing
Common alternative resource values-sw600dp is missing
Common alternative resource sw720dp-land is missing
Missing support library android-support-v11.jar
Android-manifest.xml - you should not use
    android:resizeable atribute
API loss: 0.0 %
Supports screens loss: 0.0 %
Compatable screens loss: 0.0 %
OpenGL loss: 0.0 %
Real loss: 0.0 %
Graphical experience loss: 76.9 %
```

Analizom je ustanovljeno da su testirane aplikacije kreirane tako da se u obzir uzimao problem fragmentacije i potencijalni gubitak tržišta. To potvrđuje da, prema dobijenim rezultatima, nijedna aplikacija nema procentualni gubitak tržišta. S druge strane, analizom je utvrđeno da aplikacije gotovo ne posjeduju alternativne resurse, što značajno može uticati na korisničko iskustvo prilikom upotrebe aplikacije. Isto tako primjetno je i

da se biblioteka podrške ne koristi. Na slici 2. je dat primjer lošeg grafičkog iskustva u igrici Frozen Bubble na kojoj je izražen problem mutnih i pikselizovanih grafičkih resursa.



Slika 2: *Primjer lošeg grafičkog iskustva u igrici Frozen Bubble. Od lijevo prema desno date su fotografije ekrana sljedećih rezolucija: 480 x 780 px, 1280 x 720 px i 1080 x 1920 px.*

Kreirani analizator ima više prednosti u odnosu na postojeća rješenja. Na primjer, Android Studio daje samo na osnovu minimalne verzije API-ja, koja se zahtjeva u aplikaciji, procentualnu vrijednost na koliko uređaja će aplikacija biti dostupna. Takođe, Google Play konzola za programere daje samo određene preporuke za unapređenje aplikacije i broj uređaja sa kojim aplikacija neće biti kompatabilna. Za razliku od toga predloženi analizator, pored procentualnog gubitka zbog nepodržavanja verzija API-ja, daje i procentualno gubitke zbog nepodržavanja OpenGL verzija i veličina i gustina ekrana, kao i kumulativni procentualni gubitak za ta tri skupa. Takođe, analizator daje i procentualni gubitak u vezi korisničkog iskustva, ali i prepoznaje i evidentira fragmentacije u aplikaciji te daje preporuke za njihovo otklanjanje.

7. ZAKLJUČAK

Fragmentacija Androida predstavlja jednu kompleksnu sredinu iz razloga učestvovanja mnogo proizvođača uređaja sa Android operativnim sistemom, koji personalizuju uređaje prema svome nahođenju. Iako postoje određena ograničenja, koja se odnose na program kompatabilnosti Androida, proizvođači su ipak u mogućnosti da kreiraju uređaje koji odstupaju od standarda. To nikako ne može biti negativna osobina, jer se na taj način otvaraju nova potencijalna tržišta, te uređaji koji su doskora bili mimo standarda vrlo lako postaju dio standarda.

Dinamično tržište sa brzim razvojem i velikim brojem novih proizvoda, najviše utiče na nemogućnost kreiranja mehanizama defragmentacije za sve operativne kontekste. Primjetno je da Android kreira mehanizme defragmentacije i očigledno je da se ti mehanizmi kreiraju tek kada se na tržištu stvori odgovarajući procentualni udio uređaja sa tim tipom fragmentacije. Stoga na tržištu uvijek postoje uređaji za koje ne postoji određeni tip defragmentacije. Već u trenutku pisanja zaključka kreirane su nove verzije operativnog sistema, a procentualni udio operativnih sistema, te veličina i gustina ekrana, prema statistici sa Google Play-a, se, takođe, mijenja.

Na kraju, može se zaključiti da i mnoge popularne aplikacije nisu u potpunosti ostale imune na problem fragmentacije, iako je izvršena samo pojednostavljena analiza. Stoga, postoji potreba za većim fokusom na ovaj problem te kreiranju kvalitetnog analizatora aplikacija koji bi ukazao na probleme fragmentacije u istoj.

LITERATURA

[1] *Leonid Batyuk, Aubrey-Derrick Schmidt, Hans-Gunther Schmidt, Ahmet Camtepe, Sahin Albayrak, MobileWireless Middleware, Operating Systems, and Applications - Developing and Benchmarking Native Linux Applications on Android. Springer Berlin Heidelberg, 2009.*

[2] *"Smartphone OS Market Share". IDC, http://www.idc.com/prodserv/smartphone-os-market-share.jsp (sadržaj preuzet 02.02.2015.)*

[3] *Hyung Kil Ham, Young Bom Park, Software Engineering, Business Continuity, and Education - Mobile Application Compatibility Test System Design for Android Fragmentation. Springer Berlin Heidelberg, 2011.*

[4] *Damith C. Rajapakse, Handbook of Research on Mobile Software Engineering: Design, Implementation, and Emergent Applications - Fragmentation of Mobile Applications. IGI Global, 2012.*

[5] *"Android compatability". Android, https://source.android.com/compatibility/index.html (sadržaj preuzet 04.03.2015.)*

[6] *"Filters". Android, http://developer.android.com/google/play/filters.html (sadržaj preuzet 09.03.2015.)*

[7] *"The fallacy of Android fragmentation – a statistical analysis". AndroidAuthority , http://www.androidauthority.com/the-fallacy-of-android-fragmentation-a-statistical-analysis-73646/, (sadržaj preuzet 01.03.2015.)*

[8] *"Dashboards". Android, http://developer.android.com/about/dashboards/index.html (sadržaj preuzet 04.05.2015.)*

[9] *"Google Play Services". Android, http://developer.android.com/google/play-services/index.html (sadržaj preuzet 29.05.2015.)*

[10] *Gianluca Pacchiella, Instant Android Fragmentation Management How-to. Packt Publishing, 2013.*

[11] *"Support library". Android, http://developer.android.com/tools/extras/support-library.html (sadržaj preuzet 18.03.2015.)*

[12] *"ActionBarSherlock". ActionBarSherlock, http://actionbarsherlock.com/ (sadržaj preuzet 22.03.2015.)*

[13] *"Supporting Multiple Screens". Android, http://developer.android.com/guide/practices/screens_support.html (sadržaj preuzet 07.04.2015.)*

[14] *"Providing Resources". Android, http://developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources (sadržaj preuzet 17.04.2015.)*

[15] *"Material Design", Android, http://www.google.com/design/spec/material-design/introduction.html (sadržaj preuzet 29.05.2015.)*

Nikola Obradović – magistar računarstva i informatike, Elektrotehnički fakultet Banjaluka, RS, BiH
Kontakt: nikola.obradovic@etfbl.net
Oblasti interesovanja: mobilne aplikacije, Android, kriptografija, sigurnost, objektno-orijentisano programiranje i modelovanje

prof. dr Zoran Đurić – Elektrotehnički fakultet Banjaluka, RS, BiH
Kontakt: zoran.djuric@etfbl.net
Oblasti interesovanja: sigurnost, kriptografija, PKI, platni sistemi i protokoli, formalna verifikacija, mašinsko učenje, objektno-orijentisano programiranje i modelovanje, Internet programiranje, razvoj mobilnih aplikacija, XML-bazirana međuoperativnost, Web servisi, računarske mreže, penetration testing, sistem integracija