Poboljšanje energetske efikasnosti softvera

Istraživački rad u okviru kursa
Upravljanje projektima u indutriji i nauci
Matematički fakultet

Ana Vuksić, Gorana Vučić
mi13100@alas.matf.bg.ac.rs,
mi13181@alas.matf.bg.ac.rs

22. decembar 2017

Sažetak

Uređaji koje mi koristimo imaju neki vid operativnog sistema i utiču na potrošnju energije, tako da su benefiti korišćenja ovakvih komponenti u našim proizvodima veliki. U ovom radu ćemo ispitati da li optimizacije i dobre odluke na softverskom nivou mogu povoljno da utiču na potrošnju električne energije. Pokazaćemo da razvoj informacionih i komunikacionih tehnologija (eng. *ICT*, dalje IKT) ima sve veći uticaj na ekološko i ekonomsko okruženje, a zatim ćemo dati nekoliko predloga i smernica da se smanji potrošnja električne energije kako na aplikativnom, tako i na sistemskom sloju programa.

Sadržaj

1 Uvod **2**

2 IKT i okruženje **2**
2.1 Potrošnja električne energije 2
2.2 Ekološki aspekti 2
2.3 Metrike i metode 3

3 Metodologije za energetski efikasno programiranje **4**
3.1 Efikasnost aplikativnog softvera 5
3.1.1 Računska efikasnost 5
3.1.2 Efikasnost podataka 6
3.2 Efikasnost operativnih sistema 7
3.2.1 Mehanizmi za očuvanje snage 7
3.2.2 Energetske politike (pravila) 8
3.2.3 Ponašanje koje prati okruženje uređaja 8
3.3 Opšti problemi i rešenja ili sugestije 9

4 Zaključak **10**

Literatura **10**

1 Uvod

Tokom poslednjih godina upotreba informacionih i komunikacionih tehnologija je naglo porasla u poslovnim aspektima ali i u našim svakodnevnim životima. Kako sve zavisi od električne energije porast rasprostranjenosti računara, prvenstveno onih u prenosivoj formi, neposredno utiče na porast potoršnje električne energije. Zbog ovoga su IKT postale značajan deo našeg ekosistema.

Dosadašnji napori da se smanji utisak IKT na okruženje su bili usmereni na direktnog potrošača: hardver. Razvijane su nove tehologije koje su učinile elektronske komponente štedljivije i efikasnije, ali to nikako ne utiče na to kako se taj hardver koristi. Imajući to u vidu nama je cilj da istražimo i predložimo poboljšanja i optimizacije na softverskom nivou.

2 IKT i okruženje

Da bi uopšte moglo da se priča o povećanju energetske efikasnosti potrebno je opisati kontekst unutar koga se bavimo tim pitanjem. U ovoj glavi ćemo izložiti ekonomske i ekološke aspekte, pokazati uticaj razvoja IKT i predstaviti metrike i metode za merenje energetske efikasnosti softvera.

2.1 Potrošnja električne energije

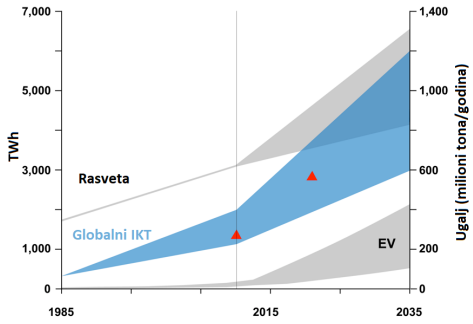
Računari i informacione tehnologije (IT) troše značajne količine električne energije i time predstavljaju značajan teret električnim mrežama [1]. IKT su odgovorne za 2% globalne potrošnje električne energije u 2007. godini [2] što je ekvivalentno godišnjoj proizvodnji osam nuklearnih elektrana [3]. Još jedan primer potrošnje električne energije se može videti na slici 1. Veliki sistemi, poput superračunara i centara podataka (eng. *data center*), su značajni potrošači. Oko 30% troškova centara podataka su od električne energije [1]. Povećana potrošnja vodi ka povećanoj složenosti u snabdevanju struje što iziskuje dodatne troškove. Veći sistemi koji zahtevaju veću količinu električne energije stvaraju i veću količinu toplote koja vodi do povećanih troškova za hlađenje [1].

Uprkos činjenici da je dosta napora uloženo u razvoj efikasnih IKT-a, dosta istraživanja i dalje pokazuje da su moguća i značajnija poboljšanja za smanjenje potrošnje električne energije [2, 4, 5, 6, 7].

2.2 Ekološki aspekti

Kako bi smanjili ekološke probleme IKT-a i stvorili održivo okruženje usmeravamo se ka „zelenim" sistemima. Termin „zelene IKT" (eng. *Green ICT*) je široko rasprostranjen i koristi se za ekološki naklonjene IKT-e. Definiše se kao izučavanje i primena ekološki održivih IT, što uključuje efikasan: dizajn, proizvodnju, upotrebu i odstranjivanje (odnosi se na fizički otpad i reciklažu) računara, servera i povezanih podsistema, i to bez ili sa minimalnim uticajem na okolinu [1]. Zelene IKT naglašavaju potrebu za smanjenjem ekološkog uticaja IKT-a tako što će se umanjiti njihovu upotrebu električne energije i emisija gasova koji doprinose efektu staklene bašte (proces kojim atmosfera čini da temperatura na površini planete bude veća nego kada nema atmosfere) [9, 10].

Potrošnja energije je blisko povezana sa emisijom ugnjenika [11]. IKT su odgovorne za oko 2% emisije gasova staklene bašte [12]. Svaki računar u



Slika 1: Globalni sistem IKT su 2010. bile odgovorne za potrošnju iste količine uglja kao i rasveta na globalnom nivou 1985. godine [8].

upotrebi emituje tonu ugnjen-dioksida na godišnjem nivou [1]. Otpuštanje ovog i drugih gasova doprinosi globalnom zagrevanju koje zauzvrat ima mnoge uticaje na okruženje. Samo neki od njih su: gašenje okeanskih struja zbog poremećaja saliniteta vode, umiranje amazonskih prašuma, topljenje permafrosta, topljenje ledenog pokrivača zapadnog Antarktika, poremećaj indijskih monsuna, topljenje ledenog pokrivača Grenlanda itd. [13].

2.3 Metrike i metode

Zarad razumevanja, merenja i poređenja energetske efikasnosti potrebne su nam metrike i metode. Ne postoje standardi za njih pa zbog toga nam je neophodno da ih mi definišemo.

Metrika se može definisati kao „sistem ili standard merenja" [14]. Ona definiše jedinicu i pruža osnovu za poređenje dva merenja. U osnovi ona može biti definisana kao [15]:

$Efikasnost = Obavljen\ Posao/Upotrebljen\ trud$

U kontekstu energetske efikasnosti možemo koristiti formulu:

$Energetska\ efikasnost = Obavljen\ posao/Upotrebljena\ energija$

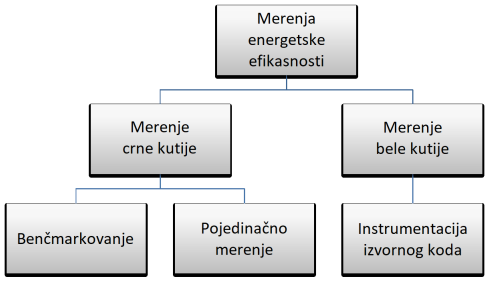
Obično „obavljen posao" i „upotrebljenu energiju" posmatramo tokom ograničenog vremenskog perioda, ali i rezultat i potrošnja energije su vezane za vremenski period što obično vodi do eliminacije vremena pri čemu dobijamo dimenziju „usluga po energiji" kao što kByte/Ws može biti za internet uslugu [12]. Jedan od razloga zašto je merenje energetske efikasnosti softvera komplikovano i ne postoje standardne metrike i metode je zato što „urađen posao" može drastično da se razlikuje od jednog softvera do drugog. To može biti: sortiranje 100000 double vrednosti u nizu ili poslati 100MB podataka preko mreže itd. Jedinice efikasnosti bi bile „sortirani brojevi/Ws" i „poslati podaci/Ws" redom. Postoji bezbroj mogućnosti što čini stvaranje standarda koji bi odgovarao svim vrstama softvera veoma teškim, a možda čak i nemogućim. Zbog toga je neophodno definisati odgovarajuće metode i metrike za pojedinačne vrste problema. Primeri nekih metrika se mogu videti u tabeli 1.

Metod je posebna procedura ostvarivanja određenog cilja koja je često sistematska ili uspostavljena. Predstavlja skup tehnika koje opisuju kako se merenje vrši.

Aspekt	Metrika
Energetska efikasnost	Energija / Jedinica posla
Intezitet CPU-a	Broj ciklusa CPU-a
Iskorišćenost memorije	Upotrebljenost memorije
Neiskorišćenost	Vreme neiskorišćenosti

Tabela 1: Primeri korisnih metrika u smislu energetske efikasnosti [15]

Jedna od sistematskih klasifikacija metoda za merenje energetske efikasnosti softvera deli ih na dve grupe: **Merenje crne kutije**, a pod njom *benčmarkovanje* (eng. *benchmarking*) i *pojedinačno merenje*, i **Merenje bele kutije** i pod njom *instrumentacija izvornog koda* (slika 2) [14].



Slika 2: Metodi za merenje energetske efikasnosti [14].

Merenje crne kutije softver posmatra kao „crnu kutiju" i ispituje efikasnost softvera kao celinu bez ikakvog znanja o individualnim komponenetama. Samim tim mogu da otkriju da li je softver neefikasan, ali ne i izvor neefikasnosti. U merenja crne kutije ubrajamo prethodno pomenute metode: *benčmarkovanje* i *pojedinačno merenje* [14]. *Benčmarkovanje* označava da se softver testira kroz niz specifičnih i unapred osmišljenih testova koji daju uvid u to kako se softver ponaša u ekstremnim slučajevima. *Pojedinačno merenje* se odnosi na to da se ponašanje softvera prati pod realnim slučajevima upotrebe i poredi različite konfiguracije softvera u tim situacijama.

Merenje bele kutije nasuprot metodima crne kutije posmatraju sâm kôd. Unutrašnji podaci mogu se uzeti kao indikatori za stvaranje metrika. Metode bele kutije su pogodnije za otkrivanje delova programa koji su veći potrošači resursa i njihovo unapređivanje [14]. *Instrumentacija izvornog koda* se odnosi na implementaciju instrukcija u kodu koje nadgledaju određene komponente sistema.

3 Metodologije za energetski efikasno programiranje

Kada govorimo o energetski efikasnom softveru razlikujemo metodologije koje se odnose na njegove dve vrste: aplikativni i sistemski softver. Za sistemski softver fokusiraćemo se na operativne sisteme. Cilj nam je više da prikažemo pregled raznih pristupa, nego da ulazimo u detalje pojedinačnih rešenja.

3.1 Efikasnost aplikativnog softvera

U nastavku navodimo razne preporuke za pisanje efikasnijeg koda. Za početak govorimo o optimizaciji aplikativnog softvera, a ne sistema kao celine. Pre nego što izvršimo bilo kakvu optimizaciju korisno je izvršiti merenja na početnoj verziji kako bismo mogli da poredimo rezultate kasnije i dobili bolju sliku o ostvarenim poboljšanjima [7]. Još jedna preporuka je ograničiti pozadinske procese i servise koje mogu uticati na izvršavanje softvera koji testiramo [7].

3.1.1 Računska efikasnost

Prosto rečeno računska efikasnost znači posao odraditi brže. Trud koji je uložen u bolje performanse softvera ne štedi samo vreme već i energiju. Ovo se zove „trka ka besposlenosti" (eng. *race to idle*) [4]. Što brže ispunimo zadatak i dovedemo računar u „besposleno" stanje (eng. *idle*) više ćemo energije uštedeti. Kako bismo ostvarili računsku efikasnost uvodimo softverske tehnike koje ostvaruju bolje performanse kao što su: efikasni algoritmi, višenitno izvršavanje, vektorizacija [4], odmotavanje petlji, optimizacije kompilatora, izbor programskog jezika [7] i dr. Ukratko ćemo opisati navedene tehnike.

Efikasni algoritmi. Izbor algoritma i strukture podataka može da napravi veliku razliku u performansama programa [4]. Preporuka za povećanje energetske efikasnosti je upravo korišćenje takvih algoritama koji brže završe posao i pre dovedu procesor u „besposleno" stanje [7]. Primer sortiranja 200000 double vrednosti iz [14] kao rezultat dobija da korišćenjem *bubble sort* algoritma imamo efikasnost od 18 sortiranih vrednosti po džulu(J), dok *heap sort* algoritam ima rezultat od 27,3 sortiranih brojeva po džulu(J). Ovo nas dovodi do zaključka da algoritmi sa boljim asimptotskim vremenom izvršavanja imaju i bolju energetsku efikasnost [14].

Višenitno izvršavanje. Izvršavanje programa može biti ubrzano iskorišćavanjem više niti i jezgara što povećava „besposleno" vreme procesora što dalje doprinosi uštedi energije [7]. Bitno je ispravno izbalansirati posao po nitima jer u suprotnom može doći do povećane potrošnje energije [7].

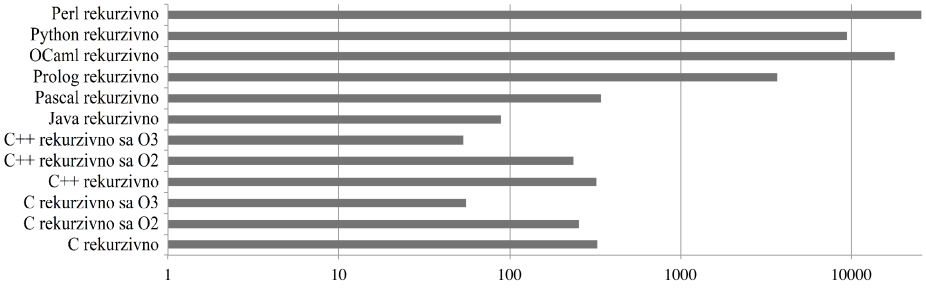
Vektorizacija. Podrazumeva korišćenje vektorskih instrukcija ili instrukcija vrste „jednostruka instrukcija, višestruki podaci" (eng. *Single instruction, multiple data, SIMD*) [16]. Vektorizacija je proces prerade petlji tako da umesto obrade jednog po jednog elementa niza N puta obradimo (recimo) 4 elementa istovremeno nad nizom N/4 puta. Ako je moguće vektorizovati naše rešenje dobićemo bolje performanse kao i odgovarajuću energetsku dobit [4].

Odmotavanje petlji može poboljšati performanse jer smanjuje dodatni posao (*overhead*) koji se javlja kod malih petlji [7]. Ovo se ostvaruje tako što se instrukcije koje se pozivaju u više iteracija kombinuju u jednu iteraciju. Sporedni efekti mogu biti povećan broj iskorišćenih registara i duži kôd [7].

Optimizacije kompilatora. Često su nam dostupne dodatne opcije za optimizaciju programa kao što su „O2" i „O3" na GNU kompilatora [7, 17, 18]. Videti sliku 3.

Izbor programskog jezika. Ako je moguće poželjno je uzeti implementaciju programskog jezika i biblioteke koje su nastrojene ka manjoj iskorišćenosti energije (eng. *idle-power-friendly*). Neki jezici višeg nivoa mogu izazvati češća buđenja u poređenju sa programskim jezicima niskog

nivoa kao što je C [7]. Videti sliku 3 gde su izvršena poređenja rekurzivne implementacije algoritma za kule Hanoja u različitim programskim jezicima po potrošnji energije u džulima [18].



Slika 3: Energetska potrošnja rekurzivne implementacije programa za kule Hanoja u različitim programskim jezicima i sa različitim optimizacijama kompilatora (koristeći skalu sa logaritamskom bazom 10) [18].

3.1.2 Efikasnost podataka

Moderni memorijski sistemi su hijerarhije keševa, magistrala i memorija [16, 19]. Prebacivanje podataka između elemenata koji ih obrađuju i memorije oduzima kako vreme tako i energiju. Kako bi se povećala efikasnost podataka, razvojni tim mora da dizajnira softverske algoritme koji minimizuju pomeranja podataka, memorijske hijerarhije koje čuvaju podatke blizu elemenata koji ih obrađuju i aplikacione softvere koji efikasno koriste keš memorije [19].

Metode za računsku efikasnost koje poboljšavaju performanse ponekad imaju samo neutralan efekat na energetsku efikasnost. Međutim, metode za poboljšanje efikasnosti podataka uvek smanjuju energetsku potrošnju [19]. Navešćemo neke primere u kojima se mogu primeniti metodi efikasnosti podataka za poboljšanje energetske efikasnosti.

Upravljanje I/O operacijama diska. Navešćemo kratak pregled analize energetskih karakteristika hard diskova tokom raznih aktivnosti kao i preporuke za optimizaciju. Analize su izvršene nad tipičnim karakteristikama učinka hard diskova (eng. *HDD*) na koje utiču broj rotacija (eng. *RPM*), vreme ponalaska podatka (eng. *Seek Time*), rotaciono kašnjenje (eng. *Rotational Latency*) i održiva brzina prenosa (eng. *Sustainable Transfer Rate*) [20].

- Kod velike količine sekvencijalnih podataka čitanje u većim komadima zahteva manju iskorišćenost procesora i manje energije. Npr. korišćenje blokova veličine 8KB ili većih može da doprinese poboljšanju performansi.
- Efektivna upotreba asinhronih I/O operacija korišćenjem NCQ-a (eng. *Native Command Queuing*) poboljšava performanse i štedi energiju. Aplikacije koje imaju neuređen redosled I/O ili I/O operacije sa više fajlova trebalo bi da koriste asinhroni I/O kako bi dobile prednost od NCQ-a. Stavićemo sve zahteve za čitanje u neki red i koristi događaje (event) i povratne pozive (callback) kako bi saznali kada je operacija čitanja završena.

- Čitanje fragmentisanog fajla u odnosu na neizdeljen je skuplje u pogledu performansi i energetske efikasnosti. Da bi se izbegle ove situacije preporučuje se da se za velike fajlove unapred alocira sva potrebna memorija prilikom njihovog kreiranja. Krajnji korisnici mogu da pomognu tako što će periodično vršiti defragmentaciju.
- Kada se više niti takmiči za I/O diska možemo ih staviti u red i osloniti se na NCQ. Preuređivanje zahteva može da pomogne, podigne performanse i uštedi energiju. Kada se više niti takmiči za disk to može da izazove šlajfovanje (eng. *disk trashing*). Ujedinjavanje svih operacija čitanja/pisanja u jednu nit će umanjiti šlajfovanje (označava pojavu da sistem troši više vremena za upisivanje i čitanje iz virtuelne memorije nego na instrukcije iz korisničkog softvera).

Dobavljanje unapred i keširanje (eng. *Pre-fetching and caching*). Istraživanje navedeno u [21] ispituje da li je moguće uštedeti energiju prilikom reprodukcije DVD-a. Rezultat eksperimenta ukazuje da je ubrzavanje diska (spin-up) energetski najzahtevnija akcija koja zahteva oko 4W tokom kratkog perioda. Trajno čitanje sa diska troši oko 2.5W i predstavlja još jednu oblast gde je moguće ostvariti uštede.

Iz rezultata istraživanja možemo da izvedemo sledeće zaključke i preporuke:

- Baferisanje. Aplikacije za reprodukciju DVD-a koje koriste tehniku baferisanja smanjuju potrošnju za DVD za 70% dok za celokupnu platformu za oko 10% u poređenju sa drugim tehnikama.
- Minimizovanje upotrebe DVD uređaja. Smanjiti ubrzavanja (eng. *spin-up*) i usporavanja (eng. *spin-down*) uređaja kao i broj pristupa za čitanje.
- Prepustiti OS-u da upravlja frekvencijom procesora. Ne preporučuje se menjanje načina rada procesora iz aplikacije.

3.2 Efikasnost operativnih sistema

U računarstvu, operativni sistem (OS) je skup programa i rutina odgovornih za kontrolu i upravljanje uređajima i računarskim komponentama kao i za obavljanje osnovnih sistemskih radnji. OS treba da izmeri ili proceni dinamičku potrošnju energije, predvidi iskorišćenost komponenti i pokrene mehanizam za čuvanje energije kako bi smanjio potrošnju i time podigao energetsku efikasnost [22].

Neki od osnovnih vidova uštede enegije i sami nesvesno koristimo, to bi bile šeme za potrošnju energije kada je uređaj na punjenju (eng. *plugged in*) ili na bateriji (eng. *battery*) [23].

3.2.1 Mehanizmi za očuvanje snage

Glavna ideja mehanizama za očuvanje snage je da upravlja stanjem sna, tj. da se isključe glavni elementi kada nema podataka koje treba primiti ili poslati, sve ovo se radi kako bi se sačuvala baterija i smanjila potrošnja.

Ima mnogo raznih mehanizama za očuvanje snage koje operativni sistem može da iskoristi. Te tehnike su uglavnom zasnovane na pretpostavci o dinamičkom utrošku snage i radu IKT sistema. Potrebna je i određena hardverska kompatibilnost koja dozvoljava da se izmeri trenutna potrošnja energije osvrtanjem na količinu posla koja se obavlja. Sve ovo zajedno igra veliku ulogu u postizanju dobrih rezultata [24].

Najšire primenjene tehnike za očuvanje energije kod operativnih sistema su:

- **Dinamičko skaliranje napona i učestalosti** (eng. *Dynamic Voltage and Frequency Scaling* - **DVFS**). Dinamičko skaliranje učestalosti (takođe poznato kao CPU regulisanje) jeste tehnika u arhitekturi računara gde se frekvencija mikroprocesora može automatski podesiti, bilo da sačuva napajanje ili da smanji količinu toplote generisanu od strane čipa. Najčešće se koristi u laptopovima i drugim mobilnim uređajima, zato što energija dolazi od baterije i na taj način je ograničena. Smanjeno je zagrevanje, pa se i ventilator manje koristi, smanjuje i buku i potrošnju energije. Često se kombinuje sa dinamičkim rasipanjem (menjanjem napona).
  Moramo da pazimo da ne usporimo previše rad samog uređaja onda će više raditi i dobijamo kontraefekat. Frekvencija CPU-a ima veliki uticaj na količinu energije. Kada nije potreban visoki performans snižavanje frekvencije će doprineti očuvanju energije.
- **Napredna konfiguracija i interfejs snage** (eng. *Advanced Configuration and Power Interface* - **ACPI**) je industrijska specifikacija za efikasno upravljanje potrebnom energijom na desktop i mobilnim uređajima. ACPI omogućava sistemu da kontroliše količinu snage koja je data svakom uređaju uključenom na računar na korišćenje. Dozvoljava operativnom sistemu da isključi periferne uređaje, kao što su monitori, hard diskovi posle određenog vremena neaktivnosti, i dr.
  Široko je rasprostranjen i deo skoro svih modernih operativnih sistema. Korisnik može i sam da podesi neke performanse. ACPI mora da bude podržan od strane matične ploče, BIOS-a i OS-a.

3.2.2 Energetske politike (pravila)

Plan potrošnje snage se sastoji od grupe različitih podešavanja pravila za potrošnju energije. Jedan od problema kod politika snage je da ih ima puno. Tako da se javlja puno „najboljih" politika, tj. onih za koje se tvrdi da su najbolje. Međutim, ne možemo za samo jednu tvrditi da je najbolja. Ima jako puno koeficijenata koje treba uzeti u obzir. Stanja i samo okruženje se dinamički menjaju. Tako da treba pažljivo birati koja se i kad koristi. Još jedna stvar na koju treba obratiti pažnju je to da postoji mnogo različitih kombinacija programa koji rade u isto vreme i koriste resurse uređaja. I razni programi mogu biti korišćeni u različitom vremenskom rasponu. Mala je šansa da će se vršiti eksperimenti za sve te mogućnosti.

Možemo zaključiti da ne postoji najbolja politika. Kod ugrađenih sistema možemo uzeti određenu politiku kako je kod njih poprilično homogena količina rada. Međutim, to ne možemo i za personalne računare zbog čestih promena u radu. Najbolje rešenje za ovaj problem je dinamička promena koja se vrši u hodu, menjanjem politika po potrebi.

3.2.3 Ponašanje koje prati okruženje uređaja

Pod ponašanjem koje prati okruženje uređaja se podrazumeva praćenje trenutnog stanja uređaja i raznih njegovih funkcija koje mogu da troše energiju, kao što su na primer WiFi i Bluetooth, na koje odlazi jako veliki procenat utrošene energije IKT-a. Ako se dobro upravlja ovim funkci-

jama može se značajno uštedeti snaga. Zato je bitno da razmišljamo o mogućnostima za njihovo poboljšanje.

Kako se telefoni sve više razvijaju i postaju kompleksniji a i dalje su ograničeni baterijom, bitno je da pokušamo da očuvamo što više snage i da se ne troši previše na pozadinski rad. Samo neke od stvari koje ovi uređaji prate dok rade su: osvetljenje, buka, kvalitet mreže... Značajna količina energije može da se uštedi pomoću njihove analize.

Primer kojim ovo možemo ilustrovati je osvetljenje ekrana na mobilnom uređaju. Ako ekran nije u upotrebi smanjuje se njegovo zračenje i time se stvara ušteda energije.

3.3 Opšti problemi i rešenja ili sugestije

Do sad smo se bavili rešavanjem pitanja specifičnih za sâm softver i platforme na kojima se on pokreće, a sada ćemo diskutovati o problemima nezavisnim od platformi i sadržaja programa.

Nasuprot uloženim naporima, istraživanja [25, 26] su pokazala da održivost i energetska efikasnost i dalje nisu dovoljno uključeni u tradicionalne modele softverskog inženjerstva. Kako bi ograničavanje samo na period razvoja softvera, u užem smislu, bilo previše uska perspektiva, ceo životni ciklus softvera mora da bude optimizovan. Jedan pristup razvoja softvera koji uključuje energetsku efikasnost kao nefunkcionalan zahtev je **Grinsoft** (eng. *GREENSOFT*) model [26]. Prvo ćemo definisati šta su nefunkcionalni zahtevi.

Nefunkcionalni zahtevi pri dizajniranju softvera su zahtevi koji se ne odnose direktno na svrhu softvera i ono čemu služi, ali koji se mogu odraziti na njegovu upotrebljivost. Na primer, GPS (Globalni sistem za pozicioniranje, eng. *Global Positioning System*) program mora da brzo i precizno određuje poziciju korisnika, ali ako se on pokreće na mobilnom uređaju poželjno je da program troši malo energije da bi mogao duže da se koristi. Stoga je prirodno opisati energetsku efikasnost kao nefunckionalan zahtev.

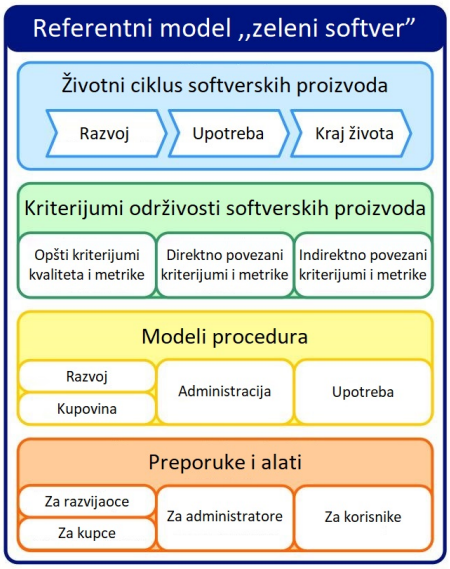
Grinsoft model je referentni konceptualni model koji sadrži smernice za celokupni životni ciklus softvera, metrike i kriterijume održivosti sofvera. Namenjen je kako razvijaocima, tako i administratorima i korisnicima softvera. On uključuje celokupne aspekte održivog razvoja u proces razvoja softvera umesto da se usredsređuje na čistu energetsku efikasnost.

Kao što se vidi na slici 4 Grinsoft se sastoji od četiri sastavna dela: životni ciklus softvera, kriterijumi održivosti softverskih proizvoda, modeli procedura i preporuke za postupke i alate.

Nasuprot tradicionalnim životnim ciklusima, Grinsoft odražava „razmišljanje životnom ciklusu" (eng. *life cycle thinking*), čiji je cilj da proceni ekološku, društvenu, ljudsku i ekonomsku kompatibilnost proizvoda tokom celog njegovog životnog ciklusa [26]. Rezultati ovih procena se mogu iskoristiti za optimizaciju proizvoda.

Komponenta „Kriterijumi održivosti softverskih proizvoda" daje modele kvaliteta i standardizovane metrike koje se mogu koristiti u bilo kojoj fazi razvoja za reviziju softvera.

Kako treba uzeti i okolnosti pod kojima se razvija tim delom se bavi komponenta „Modeli procedura". Ona uključuje modele procedura zasnovane na njihovim korisnicima: razvijaocima, kupcima, administratorima i krajnjim korisnicima. Svaki model može biti prilagođen različitim kontekstima.



Slika 4: Pregled Grinsoft modela

Na kraju, poslednja komponenta su preporuke i alati. Oni su dostupni raznim ulagačima u sistem. Sastoje se iz raznih kontrolnih lista, smernica, najboljih primera iz prakse, softverskih alata i alata opšte namene koji podržavaju ulagače prilikom primenjivanja „zelenih” i održivih tehnika.

Iako razna istraživanja preporučuju zelene modele razvoja sistema, ne postoji nijedan takav model koji je postao opšteprihvaćen industrijski standard [7]. Zato je dužnost akademskih istraživača, ali i razvojnih timova da podstiču potragu za prihvatljivijim rešenjima.

4 Zaključak

U ovom radu smo se potrudili da približimo važnost uštede energije kroz dobre prakse kako programerima, tako i ostalim osobama koje učestvuju u životnom ciklusu softvera. Kroz analizu nekoliko pristupa softverskom dizajnu ponudili smo pregled raznih problema ali i rešenja i predloga za njihovo prevazilaženje.

Pokazali smo da softver može da utiče na potrošnju energije koliko i hardver, i da čak i mala poboljšanja i prepravke mogu bitno da utiču na potrošnju električne energije. Na primer, odabir programskog jezika može da smanji potrošnju energije za red veličine, kao što smo pokazali na primeru u odeljku 3.1.1.

Iako se u polju softverskog dizajna, gledanog iz ugla energetske efikasnosti, nalaze mnogobrojne nedovoljno istražene ideje koje bi doprinele uštedi energije, ovoj temi se još uvek ne posvećuje dovoljno pažnje, nasuport hardverskoj optimizaciji energetske potrošnje, čak i ako su dobiti odmah i lako uočljive. Zato se u budućnosti treba posvetiti tome da energetska pitanja postanu nefunkcionalni zahtevi u većini pristupa dizajnu softvera. Isto tako treba dalje razrađivati konkretne programske i sistemske optimizacije unutar postojećih modela za uštedu energije.

Literatura

[1] San Murugesan, "Harnessing green IT: Principles and practices," *IT Professional*, pp. 24–33, February 2008. on-line at: https://pdfs.semanticscholar.org/eba0/69f2cb39dca7628d0cf2e4be6caaf9142e27.pdf.

[2] Andreas Winter, Jan Jelschen, "Energy-Efficient Applications," *Oldenburg Lecture Notes on Software Engineering*, April 2012. on-line at: http://www.se.uni-oldenburg.de/documents/olnse-3-2012-EEA.pdf.

[3] Aurélien Bourdon, Adel Noureddine, Romain Rouvoy and Lionel Seinturier, "PowerAPI: A Software Library to Monitor the Energy Consumed at the Process-Level," pp. 43–44, January 2013. on-line at: https://ercim-news.ercim.eu/images/stories/EN92/EN92-web.pdf.

[4] Bob Steigerwald, Abhishek Agrawal, *Developing green software*. Intel, 2011. on-line at: https://software.intel.com/sites/default/files/developing_green_software.pdf.

[5] Chris Shore, "Developing Power-Efficient Software Systems on ARM Platforms," *Technology In-Depth*, pp. 28–53, 2009. on-line at: https://pdfs.semanticscholar.org/637e/66176930cf6151590cef10c3322ab37df5d6.pdf.

[6] David J. Brown, Charles Reams, "Toward Energy-Efficient Computing," *ACM Queue*, vol. 8, February 2010. on-line at: http://dl.acm.org/ft_gateway.cfm?id=1730791&ftid=568412&dwn=1.

[7] Petter Larsson, *Energy-Efficient Software Guidelines*. Intel, April 2011. on-line at: https://software.intel.com/sites/default/files/article/57237/energy-efficient-software-guidelines-v3-4-10-11_1.pdf.

[8] Mark P. Mills, "The Cloud Begins With Coal," August 2013. on-line at: https://www.tech-pundit.com/wp-content/uploads/2013/07/Cloud_Begins_With_Coal.pdf?c761ac.

[9] *Software Engineering Aspects of Green Computing*, 2013. Symposium on Applied Computing, Online at http://trese.ewi.utwente.nl/workshops/SEAGC/.

[10] "What is the Greenhouse Effect?," 2007. on-line at : http://www.ipcc.ch/publications_and_data/ar4/wg1/en/faq-1-3.html.

[11] W. Vereecken, W. Van Heddeghem, D. Colle, M. Pickavet, and P. Demeester, "Overall ICT footprint and green communication technologies," 2010. on-line at: http://www.askind.sakarya.edu.tr/sites/askind.sakarya.edu.tr/file/Green_COMMS.pdf.

[12] L. M. Hilty, V. C. Coroama, M. Ossés, T. F. Ruddy, E. Thiébaud, "The Role of ICT in Energy Consumption and Energy Efficiency," *ICT-Ensure*, January 2009. on-line at: https://www.researchgate.net/publication/267411194_The_Role_of_ICT_in_Energy_Consumption_and_Energy_Efficiency.

[13] GeSI, "Smarter 2020," December 2012. on-line at: gesi.org/portfolio/file/5.

[14] *How to measure energy-efficiency of software: Metrics and measurement results*, 2012. First International Workshop on Green and

Sustainable Software, on-line at: https://pdfs.semanticscholar.org/e1ca/3ca384c9ae311c6f2f1b08af05f64c31d6e0.pdf.

[15] Fethullah Goekkus, "Energy Efficient Programming An overview of problems, solutions and methodologies," December 2013. on-line at: https://files.ifi.uzh.ch/hilty/t/examples/bachelor/Energy_Efficient_Programming_G%C3%B6kkus.pdf.

[16] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design, Revised Fourth Edition, Fourth Edition: The Hardware/Software Interface*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 4th ed., 2011. on-line at: http://nsec.sjtu.edu.cn/data/MK.Computer.Organization.and.Design.4th.Edition.Oct.2011.pdf.

[17] Richard M. Stallman and the GCC Developer Community, "GCC online documentation." on-line at: https://gcc.gnu.org/onlinedocs/.

[18] Adel Noureddine, Aurelien Bourdon, Romain Rouvoy, Lionel Seinturier, "A Preliminary Study of the Impact of Software Engineering on GreenIT," pp. 21–27, June 2012. on-line at: https://hal.archives-ouvertes.fr/file/index/docid/681560/filename/noureddine-bourdon-green.pdf.

[19] Dr. Guido Arnout, "Data-efficient Software and Memory Architectures are Essential for Higher Performance and Lower Power," *Information Quarterly*, vol. 4, October 2005.

[20] Karthik Krishnan, Jun De Vega, *Power Analysis of Disk I/O Methodologies*. Intel, April 2011. on-line at: https://software.intel.com/en-us/articles/power-analysis-of-disk-io-methodologies-1.

[21] Tareq H. Darwish, Rajshree Chabukswar, Kiefer Kuah, Bob Steigerwald, "HD Video Playback Power Consumption Analysis," March 2008. on-line at: https://software.intel.com/sites/default/files/34/2a/1334.

[22] Clemens Lang, "Components for Energy-Efficient Operating Systems," 2013. on-line at: https://wwwcip.informatik.uni-erlangen.de/~sicslang/ss13/akss/paper.pdf.

[23] Novell, *ZENworks 11 Configuration Policies Reference Power Management Policy*, 2016. on-line at: https://www.novell.com/documentation/zenworks114/zen11_cm_policies/data/br5knmk.html.

[24] D. C. Snowdon, E. Le Sueur, S. M. Petters, and G. Heiser, "Koala: A Platform for OS-level Power Management," pp. 289–302, ACM, 2009. on-line at: https://www.researchgate.net/publication/221351699_Koala_a_platform_for_OS-level_power_management.

[25] S. S. Mahmoud and I. Ahmdad, "A Green Model for Sustainable Softver Engineering," 2013. on-line at: https://www.researchgate.net/publication/285762532_A_green_model_for_sustainable_software_engineering.

[26] S. Naumann, M. Dick, E. Kern, and T. Johann, "The greensoft model: A reference model for green and sustainable software and its engineering," on-line at: http://www.academia.edu/6573845/The_GREENSOFT_Model_A_reference_model_for_green_and_sustainable_software_and_its_engineering.