Razvoj programskog jezika Eiffel

Seminarski rad u okviru kursa
Dizajn programskih jezika
Matematički fakultet

Jovana Adžić
jovanaadzic42@gmail.com

6. januar 2020

Sažetak

U ovom radu predstavljeni se osnovni koncepti razvoja programskog jezika Eiffel. Opisani su i jezici koji su uticali na razvoj datog: Ada, Simula, Z. Iznete su osnovne informacije o datim jezicima, kao i o jezicima koji su indirektno uticali na nastanak Eiffel-a, njihov razvoj i sam uticaj na razvoj jezika predstavljenog ovim radom . Takođe, prikazano je i razvojno stablo koje uključuje date jezike.

Sadržaj

1 Uvod

U novije vreme dolazi do sve bržeg razvoja programskih jezika. Naime, javlja se potreba primene računarstva u svim oblastima ljudskog života. Shodno tome potrebno je prilagoditi programske jezike efikasnoj primeni u svakodnevnom životu. Zbog toga je važan brz razvoj, ali i razvoj jednostavnog i ponovnog koristivog koda. Upravo sa tim ciljevima, došlo je do razvoja programskog jezika Eiffel, koji je svojim nastankom uticao na jezike koji su danas primenjivi u mnogim oblastima ljudskog života. Eiffel je programski jezik, kao i razvojno okruženje koje je najvećim delom nastalo kako bi omogućilo programerima pravljenje pouzdanih, ponovno koristivih modela softvera. Baziran na objektno orijentsianoj paradigmi, Eiffel podržava višestruko nasledjivanje, polimorfizam, generičko programiranje, enkapsulaciju podataka. Izvan svih jezičkih aspekata, Eiffel promoviše i metod softverske konstrukcije kombinujući višekratne i fleksibilne module [7]. Takođe zbog jednostavnosti koda, važna je njegova primena prilikom kompilacije jer kompilatori za računarske probleme napisani u jeziku Eiffel pružaju izvanredne tehnike optimizacije. Međutim, zbog ubrzanog napretka drugih programskih jezika nastalih pod uticajem Eiffel-a i mogućnosti njihove široke primene u raznim oblastima ljudskog života, kao i same ograničene primene Eiffel-a, njegovo korišćenje i primena su smanjene [1]. Logo programskog jezika Eiffel je dat na slici 1.

Slika 1: **Logo programskog jezika Eiffel**

2 Eiffel

Eiffel je objektno orijentisani programski jezik koji je stvoren od strane **Bertrand Majera** i **Ajfel softvera**. Majer je 1985. osmislio jezik u cilju kreiranja pouzdanijeg komercijalnog softvera; prva verzija je postala dostupna **1986**. godine. 2005. Eiffel je **ISO-standardizovan**. Dizajn samog jezika je blisko povezan sa metodom Ajfel programiranja [1]. Tačnije, oba se baziraju na određenom skupu principa, uključujući dizajn po ugovoru, odvajanje komande upita, otvoreno-zatvorene principe, princip jedinstvenog pristupa.

Definicija Eiffel jezika je internacionalni standard ISO-a. Trenutna verzija standarda iz juna 2006 sadrži neke nedoslednosti . ECMA komitet još nije proglasio bilo koju vremensku liniju i direkciju kako rešiti te nedoslednosti [9]. Eiffel-ov sistem ili program je kolekcija klasa. Iznad nivoa klasa, Eiffel definiše **klaster** koji je suštinski grupa klasa, i moguće podklaster (ugnježdeni klasteri). Klasteri nisu jezička sintaksna konstrukcija, već standardna organizaciona konvencija. Tipično Eiffel program će biti organizovan sa svakom klasom u različitim fajlovima, i svaki klaster u direktorijumu sadržaće fajlove klase. U ovoj organizaciji, podklasteri su poddirektorijumi. Klasa sadrži karakteristike, koji su slični rutinama, članovima, metodama u objektno orijentisanom programiranju. Klasa takođe definiše njene invarijante, i sadrži druge osobine, kao što

```
class
    HELLO_WORLD
create
    make
feature
    make
        do
            print("Hello␣world!:)␣%N")
        end
end
```

Listing 1: Eiffel

Slika 2: **"Hello world"u Eiffel-u**

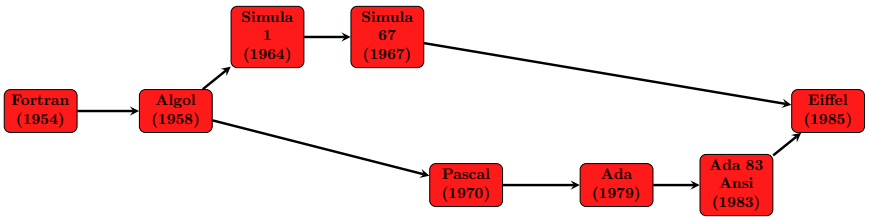
su 'beleške' sekcija za dokumentaciju i metapodatke [10]. Eiffel-ovi standardi tipa podataka, kao što su **integer**, **string**, **array** su sami od sebe klase. Svaki sistem mora imati klasu označenu kao "koren". Eiffel ima 5 klasičnih instrukcija izvršenja: dodela, kreiranje objekta, poziv rutine, uslov i ponavljanje. Tipovi atributa ili parametara klasa su od strane udruženja predložene kao zamene tipova u mnogim slučajevima. Specijalni tipovi koji se koriste prilikom rekurzije ne mogu biti zamenjeni, i tako se pojavljuju potpuno nove konstrukcija u objektno orijentisanom programiranju [10]. Jedna od posledica stvaranja ovakve konstrukcije je da Eiffelova hijerahija usaglašenosti predstavlja pravi podskup od hijerarhije nasleđivanja.

Eiffel ima značajnu primenu u reaktivnom programiranju. Jedan od glavnih razloga prodora mnogih objektno orijentisanih jezika u savremenom svetu je mogućnost ujedinjenja modula i različitih tipskih aspekata u jednu konstrukciju: klasu [1]. Prateći ovu ideju, Eiffel uvodi aktivne objekte koje mi zovemo procesima: oni objedinjuju klasu i zadatak, objekat i proces. Međutim, nisu svi objekti procesi: tu takođe postoje pasivni objekti koji moraju biti pridruženi kontrolnoj niti,to jest proces koji može biti izvršen. Naš model koristi nasleđivanje za strukturiranje konkurentnih procesa: proces je instanca klase nasleđena direktno ili indirektno iz samog PROCESA [2]. S obzirom na sekvencijalnu klasu, eksportovanjem nekih rutina, mi možemo da definišemo paralelnu klasu(proces).

Značajna primena Eiffel-a je u softverskom inženjerstvu. Njegov najveći značaj u ovoj oblasti je dizajn po ugovoru, u kome su potvrde, preduslovi, postuslovi i varijante klasi napravljene kako bi osigurale tačnost programa, a da pritom ne žrtvuju efikasnost. Naime, ugovori utvrđuju šta mora biti tačno pre nego što je rutina izvršena (preduslov) i šta se mora čekati da bude tačno nakon što se rutina završi(postuslov). Eiffel formalno podržava apstrakne tipove podataka. Javlja se mogućnost konverzije različitih tipova podataka, dok je obrada izuzetaka bazirana na dizajnu po ugovoru. Takođe, omogućava automatsko upravljanje memorijom [7]. Eiffel je čist objektno orijentisani jezik, ali daje otvorenu arhitekturu za povezivanje sa spoljnim softverom u bilo kom drugom programskom jeziku. Na primer, moguće je programirati mašinu i nivo operacija operativnog sistema u C-u. Eiffel pruža jedinstveno povezivanje sa C rutinama. Iako ne postoji direktna veza između C-a i Eiffel-a, mnogi Eiffel-ovi kompilatori izbacuju C izvorni kod kao međujezik, da se opovrgne C kompilator, za optimizovanje i portabilnost [9].

3 Razvojno stablo

Na nastanak jezika Eiffel je uticaj imalo više jezika od kojih su najznačajniji **Ada** i **Simula** . Njegov dizajn je međutim najviše povezan sa modelom Ajfel programiranja. Mnogi koncepti koji su pre svega bili predstavljeni preko ovog programskog jezika su svoju primenu kasnije našli u jezicima kao što su Java ili C. Razvojno stablo jezika Eiffel je prikazano na slici 3.



Slika 3: Razvojno stablo jezika Eiffel

3.1 Ada

ADA je programski jezik koji je pre svega zasnovan na **Paskalu**. Njega je stvorilo Ministarstvo Sjedninjenih Američkih država sedamdesetih godina prošlog veka u želji da bude primarni jezik ovog ministarstva.

Ime je dobio po **Adi Bajron**, kćerki engleskog pesnika lorda Bajrona. Ona je kao asistentkinja Čarlsa Bedidža, došla do otkrića analitičke mašine, koja se smatra prvim mehaničkim računarom u devetnaestom veku, a sama Ada prvim programerom.

Ada je dizajnirana kao odgovor na zahtev da se napravi zajednički jezik višeg nivoa za sve odbrambene aplikacije. Ada je prvobitno dizajnirana tako da ne zahteva objektno orjentisanu arhitekturu [6]. Praktično, celokupna podrška za objektno orijentisano programiranje u Adi je statička. Na primer, provera tipova može biti izvedena u vreme kompilacije.

Bitno svojstvo Ade je "multitasking"ili "multiheading".[1] Ada je uticala na stvaranje sledećih jezika: **ALGOL 68** , **Pascal** , **C++**, **Smalltalk**, **Java**. Čini se da Ada, kao jezik koji ne pripada prvenstveno objektno orijentisanoj paradigmi nije mogao kao takav u velikoj meri da utiče na potonje jezike koji pripadaju datoj paradigmi. Međutim, njegov uticaj na Eiffel se ogleda pre svega u sintaksičkoj strukturi - elementarne konstrukcije (izrazi, naredbe za kontrolu toka itd.) su definitivno uticali na odgovarajuće konstrukcije u jeziku Eiffel. Sintaksa je jednostavna i čitljiva. Na primer 'if x == 0 then y := 0; end if;' dakle, nepravilno bi bilo 'if x == 0 then y := 0; ' završetak mora biti sa 'end if;' , koja se prenela na sam jezik Eiffel, uz male izmene.

3.2 Simula

SIMULA je predstavlja ime za dva programska jezika: Simula 1 i Simula 67, koji su nastali 60-ih godina 20.veka u Norveškom kompjuterskom centru u

[1] Multitasking je podrazumeva mogućnost istovremenog izvršavanja većeg broja programa od stane jednog korisnika.

Oslu, od strane **Ole - Johana Dala** i **Kristena Najgarda**. Naime, Simula je jezik dizajniran da olakša formalni opis izgleda i pravila operacija koje se primenjuju na sisteme sa diskretnim događajima(promene stanja) [3]. Simula je istinski produžetak **Algola**, to jest sadrži Algol 60 kao svoj podskup. Kao programski jezik, pored simulacija, Simula ima opsežnu listu za obradu objekata i uvodi proširen koncept korutina u jezicima visokog nivoa.

Najgard je počeo da piše računarske programe za simulacije 1957. godine. Uočio je potrebu za boljim načinom da se opišu heterogemost i operacije sistema. Da bi nastavio dalje sa svojim idejama, Najgard je shvatio da mu je potreban neko sa boljim veštinama programiranja. Ole-Johan Dal mu se pridružio u januaru 1962. godine. Nedugo nakon toga odlučeno je da jezik bude povezan sa jezikom Algol 60 [1]. Do maja iste godine glavni koncepti za simulacioni jezik bili su postavljeni. Nastao je jezik **Simula 1**, programski jezik specijalne namene za simulaciju diskretnih događaja.

> *"The proper language development was a result of a close cooperation between Nygaard and myself, whereas Implementation conSiderations were mainly my responsibility."*[2]

Kasnih 60-ih i ranih 70-ih postojale su 4 glavne implementacije Simule: - UNIVAC 1100, - System/360 i System/370, - CDC 3000, - TOPS-10.

Ove implementacije su ugrađene na mnogim platformama. Simula 87 je najnoviji standard i rasprostranjen je na više plaformi.

Može se slobodno reći da je Simula prvi objektno orijentisani programski jezik. Kao što mu i samo ime kaže, dizajniran je za simulacije i potrebe tog domena su obezbedile okvir za mnoge karakteristike današnjih objektno orijentisanih jezika [3]. Simula se koristi za simulaciju VLSI dizajna, modeliranje procesa, protokola, algoritama, ali i u drugim oblastima kao što su: računarska grafika, podešavanje tipova i edukacija.

Simula 64 je uveo objekte, klase, nasleđivanje, potklase, simulacije diskretnih događaja i sakupljanje otpadaka. Ona je takođe uvela i karakter i tekst kao nove tipove podataka [8]. U cilju rešavanja velikih problema pristupa se razbijanju tog problema na manje zasebne celine koje se dalje mogu rešavati pojedniačno. Dekompozicija je posebno važna ukoliko se rešavanjem problema bavi više od jednog programera. Pojam klasa ovde uveden prvi put, preneo se na jezik Eiffel i tamo dodatno usavršio i postao glavni pojam samog jezika. Kao strogo tipiziran jezik, Eiffel je tipove podataka nasledio iz Simule, kao na primer string. Jedno od bitnijih svojstava Eiffel-a je mogućnost višestrukog nasleđivanja, koje je preuzeto od jezika Simula.

Na slici 4 se nalaze 'uvodni' i osnovni kodovi za jezike Simula i Ada.

3.3 Algol

Algol je jezik koji je indirektno imao dosta uticaja na nastanak programskog jezika Eiffel; veliki je uticaj ima da nastanak direktnih predaka ovog programskog jezika: Adu i Simulu. Algol, skraćeno od ALGOrithmic Language, je imperativni, proceduralni, strukturirani programski jezik čije su verzije izlazile 50-ih i 60-ih godina prošlog veka [4]. Prva verzija algola je nastala **1958**. godine. Postoje tri glavne verzije algola: algol 58, algol 60 i algol 68. Postoji i

[2] "The Roots of Objected Orientation:The Simula Language/Introduction", Ole John Dahl

```
Begin
   OutText ("Zdravo␣svete!␣:)");
   Outimage;
End;
```

Listing 2: Simula

```
with Ada.Text_IO;

procedure Zdravo is
begin
  Ada.Text_IO.Put_Line("Zdravo␣svete!␣
    :)");
end Zdravo;
```

Listing 3: Ada

Slika 4: **"Hello world"u Simuli i Adi**

još jedna verzija koju je razio Niklaus Virt, algol W, kao naslednika algola 60. Druge verzije su: algol Y, algol N, 68C, 68G . Razvoj algola 58 je počeo nakon što se grupa informatičara iz Evrope i Amerike sastala u institutu tehnologije u Cirihu [4]. Algol 68 je poslednji put prerađen 1973. godine. Algol je dizajniran da bi se izbegli neki problemi sa Fortranom. Imao je i dosta uticaja na ostale programske jezike kao što su **C**, **Paskal**, **Simula**, **Ada** i mnoge druge. Simula, jezik nastao na osnovama Algola, kao takav ima dosta konceptualnih sličnosti sa njim, koje su kasnije prenete na Eiffel. Algol program (blok) specifira niz operacija nad podacima lokalnim za sam program, jednako dobro kao i sama struktura podataka [8]. Simula nasleđuje Algol uvodeći pojmove kolekcije u programe, nazvane "procesima", koji se konceptualno izvršavaju paralelno. Ti procesi se prenose na programski jezik Eiffel -u reaktivom programiranju se paralelni procesi koriste kako bi ujedinili module(objedinili klase), što je generalno važno za samu objektno orijentisanu paradigmu.

4 Zaključak

U ovom tekstu su prikazane osnove razvoja programskog jezika Eiffel. Predstavljeni su jezici koji su najviše uticali na njegov nastanak i sam razvoj, direktno ili indirektno i predstavljeno je njegovo razvojno stablo. Za dodatne informacije posetite zvaničnu stranicu programskog jezika Eiffel i Ajfel softvera [5].

Literatura

[1] M. Bray and E. Denert. *Software Pioneers*. Springer-Verlag Berlin Heidelberg, 2002.

[2] Denis CaroInel and Yves Roudier. *Reactive Programming in Eiffel*. University of Nice, 1995.

[3] Ole John Dahl and Kristen Nygaard. SIMULA an ALGOL-Based Simulation Language. http://simula67.at.ifi.uio.no/Archive/artikkel1966cacm.pdf.

[4] D.Neel. *Tools and notions for program construction*. Syracuse University, New York, October 1982.

[5] Eiffel. Zvanična stranica Eiffel softvera. https://www.eiffel.com/.

[6] G.D.Buzzardand and T.N. Mudge. Object-based computing and the ada programming language. *Coder Stories*, March 1985.

[7] Bertrand Mayer. *Eiffel: Programming for reusability and extendibility*. ACM Sigplan Notices, 1987. http://se.inf.ethz.ch/~meyer/publications/acm/eiffel_sigplan.pdf.

[8] Bjorn Myhrhung Ole-Johan Dahl and Kristen Nygaard. *Common base language*. Norweigan Computing Center, October 1970. http://web.eah-jena.de/~kleine/history/languages/Simula-CommonBaseLanguage.pdf.

[9] ECMA 367 Standard. *Eiffel Analysis, Design and Programming Language*. June 2005. http://se.ethz.ch/~meyer/ongoing/etl/EIFFEL-STANDARD.pdf.

[10] W.R.Cook. *A Proposal for Making Eiffel Type-safe*. Hewlwtt-Packard Laboratories, April 1989.