Korektnost potklase

Aleksandar D. Kupusinac, Dušan T. Malbaški, *Member, IEEE*

***Sadržaj* — Korektnost predstavlja jedan od primarnih kriterijuma kvaliteta softverskog sistema. Ako se razmatranje usmeri na objektno orijentisane programske jezike, tačnije na problem korektnosti klase dolazi se do pojma invarijante klase. U radu je data definicija i osnovne osobine invarijante klase, koja zauzima centralno mesto kad se govori o korektnosti objektno orijentisanog programa. Zatim je data definicija korektnosti potklase. Ukazano je na pojedine probleme i na samu primenu pojma invarijante klase i time dat doprinos istraživanju opštih principa proizvodnje kvalitetnog softverskog sistema.**

***Ključne reči* — korektnost, invarijanta klase, korektnost klase i potklase, objektno orijentisano programiranje.**

I. UVOD

RAZVOJ teorijskih i praktičnih metoda koje mogu da poboljšaju kvalitet softverskog sistema ima veliki značaj za softverski inženjering koji podrazumeva proizvodnju kvalitetnog softverskog sistema. Kriterijumi kvaliteta softverskog sistema su mnogobrojni i mogu se podeliti na dve grupe: unutrašnje i spoljašnje kriterijume [1]. Spoljašnji kriterijumi (brzina, jednostavnost korišćenja softvera i sl.) obično su predmet interesovanja krajnjih korisnika, koji nisu profesionalni programeri. Međutim, profesionalnog programera više će interesovati unutrašnji kriterijumi kvaliteta (višekratna upotreba, proširivost, robustnost, korektnost i sl.), jer ključ za postizanje kvaliteta po spoljašnjim kriterijumima, leži u postizanju kvaliteta po unutrašnjim kriterijumima. Drugim rečima, da bi krajnji korisnici mogli da uživaju u vidljivim kvalitetima jednog softverskog sistema, potrebno je da projektanti i programeri prethodno obezbede skrivene kvalitete tog softverskog sistema, što podrazumeva postizanje kvaliteta po unutrašnjim kriterijumima.

Prisutan je stalni sukob među raznovrsnim kriterijumima kvaliteta, stoga je neophodno da se profesionalni programer odluči za optimalnu varijantu i napravi kompromis između sukobljenih kriterijuma kvaliteta u zavisnosti od namene softverskog sistema koji projektuje. Ma koliko bilo neophodno da se takvi kompromisi prave, postoji jedan kriterijum kvaliteta koji predstavlja izuzetak, a to je korektnost. Ne postoji opravdanje za profesionalnog programera koji zbog postizanja boljeg kvaliteta na osnovu drugih kriterijuma dovede u pitanje korektnost softverskog sistema, pa se zbog toga može zaključiti da korektnost predstavlja jedan od primarnih kriterijuma kvaliteta softverskog sistema. Fokusirajući razmatranje specijalno na korektnost objektno orijentisanog programa, odnosno na razmatranje korektnosti klase dolazi se do pojma invarijante klase. Shodno tome, u ovom radu će biti razmotren pojam korektnosti softverskog sistema i pojam invarijante klase, a zatim razmotrene i osnovne osobine. Zatim, biće ramotrena veza klasa nasleđivanje i definisana korektnost potklase, kao i pojedini problemi i primena invarijante klase u nasleđivanju, s ciljem da se ukaže na značaj istraživanja opštih principa proizvodnje kvalitetnog softverskog sistema.

A. D. Kupusinac, Fakultet tehničkih nauka, Trg Dositeja Obradovića 6, 21000 Novi Sad, Srbija (telefon: 381-21-4852441; faks: 381-21-6350727; e-mail: sasak@uns.ns.ac.yu).

D. T. Malbaški, Fakultet tehničkih nauka, Trg Dositeja Obradovića 6, 21000 Novi Sad, Srbija; (e-mail: malbaski@uns.ns.ac.yu).

II. KOREKTNOST

Sposobnost neke programske jedinice (program, klasa, potprogram, metoda) da funkcioniše prema svojoj specifikaciji, pri čemu pod specifikacijom se podrazumeva precizan opis šta data programska jedinica treba da radi, naziva se korektnost. Neka je data programska jedinica P i neka je X skup ulaza, a Y skup izlaza u najširem smislu. Specifikacija S date programske jedinice povezuje ulaze sa izlazima i zapravo je binarna relacija između skupova ulaza X i izlaza Y, odnosno $S \subseteq X \times Y$. Programska jedinica P u opštem slučaju takođe je relacija između skupova X i Y koja najčešće ima prirodu funkcije. Domen programa obuhvata samo one ulaze za koje program terminira. Korektnost programske jedinice P se može definisati na sledeći način: za programsku jedinicu P se kaže da je korektna u odnosu na specifikaciju S ako i samo ako važi $dom(P \cap S)=dom(S)$, što znači da za svaki ulaz program terminira i generiše izlaz predviđen specifikacijom [2].

Bilo koja programska jedinica, sama za sebe, nije ni korektna ni nekorektna, pa se može zaključiti da je korektnost relativan pojam. Programska jedinica je korektna, odnosno nekorektna samo u odnosu na neku specifikaciju. Postoje razni načini za zadavanje specifikacije, a jedan od najpoznatijih je pomoću formula totalne i parcijalne korektnosti. Neka se posmatra bilo koja sintaksna jedinica S (deo naredbe, naredba, blok naredbi, potprogram, metoda, program), preduslov P i postuslov Q, koji su po prirodi predikati i koji u različitim stanjima date sintaksne jedinice imaju vrednosti $\top$ i $\bot$. Za datu sintaksnu jedinicu S preduslov određuje stanje u kojem otpočinje izvršavanje S, a postuslov stanje u kojem se S završava. Stanje sintaksne jedinice određuju trenutne vrednosti njenih promenljivih. Formula totalne korektnosti glasi: ako sintaksna jedinica S otpočinje u stanju koje zadovoljava preduslov P, tada S terminira i stanje po završetku S zadovoljava predikat Q, što se simbolički prikazuje

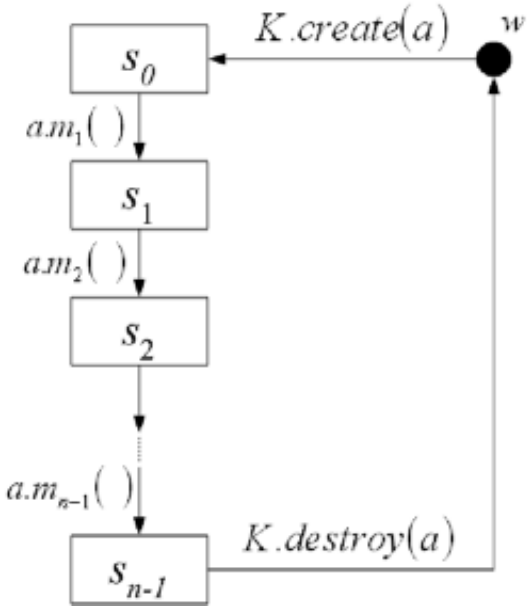
$\{P\}S\{Q\}$. Formula parcijalne korektnosti glasi: ako sintaksna jedinica S otpočinje u stanju koje zadovoljava preduslov P i ako S sigurno terminira, tada stanje po završetku S zadovoljava predikat Q, što se simbolički prikazuje $P\{S\}Q$. Obe formule su zapravo predikati i poznatije pod nazivom Hoare-ovi tripleti, u čast njihovog tvorca C.A.R. Hoare-a [3, 4], a u literaturi je zastupljenija formula totalne korektnosti, pa će o njoj biti više reči.

Prethodno razmatranje se može fokusirati na korektnost potprograma. Ako je specifikacija nekog potprograma f uređen par (P,Q), gde je P preduslov, a Q postuslov, tada je potprogram korektan ako i samo ako je tačan predikat $\{P\}f\{Q\}$. Primera radi, neka je dat potprogram $sort(a,n)$, koji služi za sortiranje po rastućem redosledu niza realnih brojeva a dužine n, tada je specifikacija potprograma uređen par (P,Q), gde je predikat $P\equiv(a$ je niz realnih brojeva dužine $n)\wedge(n\geq 2)$ preduslov, a predikat $Q\equiv a_i\leq a_{i+1}$, $i=1,2,\ldots,n\text{-}1$ postuslov. Potprogram je korektan u odnosu na specifikaciju (P,Q) ako i samo ako je predikat $\{P\}sort(a,n)\{Q\}$ tačan.

III. KOREKTNOST KLASE

Korektnost realizacije klase razmatra se sa nivoa pojedinačnih metoda i sa nivoa cele klase. Razmatranje korektnosti metoda skoro da je identično sa razmatranjem korektnosti potprograma, s tim da određena razlika proističe iz nesamostalnosti metoda, što implicira da preduslovi, odnosno postuslovi svake metode moraju da izraze relacije između tih podataka članova. Zbog toga postoji potreba za izražavanjem globalnih svojstava objekata, onih koje sve metode moraju da sačuvaju, a takva svojstva čine invarijantu klase. Naime, svi objekti iste klase mogu imati različite podatke članove, ali relacije između tih podataka članova moraju ostati iste. Invarijante mogu takođe da izražavaju semantičke relacije između metoda, ali i između metoda i podataka članova.

Neka je Σ skup svih stanja objekta ili prostor stanja, odnosno $\Sigma=\{\sigma_i \mid i=1,2,...,N\}$. Stanje u kojem se objekat nalazi određuju trenutne vrednosti njegovih podataka članova u "stabilnim" trenucima, a to znači neposredno po stvaranju objekta (početno stanje) i neposredno posle izvršenja svake metode. Sl. 1. ilustruje kretanje objekta a u prostoru stanja. Stanje nepostojanja u kojem objekat a ne postoji označeno je sa w. Objekat a nastaje metodom $a.create()$, posle čega se nalazi u početnom stanju $s_0\in\Sigma$. Izvršavanjem metoda $a.m_1()$, $a.m_2()$ itd. objekat prelazi u različita stanja $s_1,s_2,...,s_n\in\Sigma$. Objekat a se uništava metodom $a.destroy()$, posle čega se ponovo nalazi u stanju nepostojanja w. Stanja $s_0,s_1,s_2,...,s_n$ su proizvoljni elementi skupa svih stanja Σ i određuju putanju kretanja objekta a kroz prostor stanja. Na osnovu specifikacije klase skup svih stanja Σ se može podeliti na skup dozvoljenih stanja Σ_{DS} i skup nedozvoljenih stanja Σ_{NS}. Invarijanta klase I je svaki predikat koji ima vrednost T u svakom dozvoljenom stanju svakog objekta date klase, odnosno $(\forall\sigma_i\in\Sigma_{DS})$ $I(\sigma_i)=\mathsf{T}$. Stroga invarijanta klase I_K je predikat koji ima vrednost T u svakom dozvoljenom stanju svakog objekta date klase, odnosno $(\forall\sigma_i\in\Sigma_{DS})$ $I(\sigma_i)=\mathsf{T}$, a vrednost $\perp$ u svakom nedozvoljenom stanju svakog objekta date klase, odnosno $(\forall\sigma_i\in\Sigma_{NS})$ $I(\sigma_i)=\perp$ [2]. Svaka stroga invarijanta je i invarijanta klase, a obrnuto ne važi. Stroga invarijanta klase jednoznačno opisuje skup dozvoljenih stanja, tj. $\Sigma_{DS}=\{\sigma_i \mid I_K(\sigma_i)=\mathsf{T}\}$, gde je $i=1,2,...,N$, ali pošto se jedan isti skup može opisati pomoću ekvivalentnih predikata, može se zaključiti da stroga invarijanta klase je jedinstvena do nivoa ekvivalencije, odnosno sve stroge invarijante su ekvivalentne. Na primer, skup $A=\{1,2,3,4,5\}$ može se opisati kao $A=\{x|(x\in N)\wedge(x\leq 5)\}$, gde je N skup prirodnih brojeva, ali i kao $A=\{x|(x\in Z)\wedge(1\leq x\leq 5)\}$, gde je Z skup celih brojeva [5].



Sl. 1. Kretanje objekta a u prostoru stanja.

Prilikom određenja invarijante klase treba voditi računa i o tome da invarijanta klase može u sebi imati redundantni deo, koji sa jedne strane ne menja njene osobine i ne donosi ništa novo, a sa druge strane nepotrebno je usložnjava. Na primer, skup $A=\{1,2,3,4,5\}$ se može opisati kao $A=\{x|(x\in N)\wedge(1\leq x\leq 5)\}$, ali kada je već rečeno da je $x\in N$, onda je nepotrebno reći da je $1\leq x\leq 5$, jer svaki prirodan broj je veći od nule, stoga je dovoljno reći $x\leq 5$.

Ako se klasa posmatra kao model klase entiteta, onda je očigledno da pravi smisao invarijante klase jeste da dosledno očuva unutrašnje relacije koje važe između atributa svakog konkretnog entiteta iz date klase entiteta [2]. Na primer, svaki konkretan entitet «trougao» ima tri stranice koje mogu imati različite dužine stranica, ali su sve dužine pozitivni realni brojevi (jer nepostoji trougao koji ima stranicu negativne dužine) i za njegove stranice važi da je zbir dužina svake dve stranice veći od dužine treće stranice. Ako se projektant odluči da modeluje klasu entiteta «Trougao», tako što će za relevantne osobine uzeti dužine stranica, pa će klasa (objekata) *Trougao* imati tri podatka člana a, b i c, tada predikati kao što su $I_1\equiv(a>0)$, $I_2\equiv(b>0)$, $I_3\equiv(c>0)$, $I_4\equiv(a+b>c)$, $I_5\equiv(b+c>a)$ $I_6\equiv(c+a>b)$ itd. su invarijante klase. Stroga invarijanta klase Trougao biće predikat $I_K\equiv I_1\wedge I_2\wedge I_3\wedge I_4\wedge I_5\wedge I_6$. Sad se može zaključiti

da se nijedan objekat klase ne sme nalaziti u nedozvoljenom stanju, tj. u stanju u kojem je vrednost stroge invarijante klase $\perp$, jer bi to značilo da takav objekat predstavlja model entiteta koji ne postoji. Dakle, stroga invarijanta klase obezbeđuje da svaki objekat bude u dozvoljenom stanju, a to znači da bude verodostojan model entiteta (koji može postojati u domenu problema).

Već je rečeno da razmatranje korektnosti metoda skoro da je identično sa razmatranjem korektnosti potprograma, s tim da određena razlika proističe iz nesamostalnosti metoda, što implicira da preduslovi, odnosno postuslovi svake metode moraju se modifikovati. Metode za kreiranje (konstruktor) i uništavanje objekata (destruktor) specifične su po tome što konstruktor stvara objekat i iz stanja nepostojanja dovodi ga u početno stanje, a destruktor ga uništava i iz nekog stanja dovodi ga u stanje nepostojanja. Može se uvesti predikat $W \equiv$"Objekat se nalazi u stanju nepostojanja". Ako je c konstruktor date klase sa preduslovom P_c i postuslovom Q_c, a d destruktor sa preduslovom P_d i postuslovom Q_d i ako je I_K stroga invarijanta klase, tada je konstruktor korektan ako i samo ako je predikat $\{W \wedge P_c\} c \{Q_c \wedge I_K\}$ tačan, a destruktor ako i samo ako je predikat $\{P_d \wedge I_K\} d \{Q_d \wedge W\}$ tačan. Promene stanja i kretanje objekta kroz prostor stanja izvodi se posredstvom ostalih metoda, stoga za sve ostale metode važi da se njihovim preduslovima i postuslovima mora očuvati stroga invarijanta klase, jer stroga invarijanta klase mora biti zadovoljena u svakom stanju u kojem se objekat nalazi. Ako je m metoda date klase sa preduslovom P_m i postuslovom Q_m i ako je I_K stroga invarijanta klase, tada je metoda korektna ako i samo ako je predikat $\{P_m \wedge I_K\} m \{Q_m \wedge I_K\}$ tačan. Sada se može doneti zaključak o korektnosti cele klase u celini. Neka je I_K stroga invarijanta klase, tada je data klasa korektna ako i samo ako su tačni predikati:

- $\{W \wedge P_c\} c \{Q_c \wedge I_K\}$, za svaki konstruktor c,
- $\{P_d \wedge I_K\} d \{Q_d \wedge W\}$, za destruktor d,
- $\{P_m \wedge I_K\} m \{Q_m \wedge I_K\}$, za svaku metodu m.

IV. KOREKTNOST POTKLASE

Višekratna upotreba programa nije samo puko korišćenje gotovih modula, već zahteva određena podešavanja: izmene algoritma, dodavanje algoritma i podataka, pa čak i modifikaciju podataka. Mehanizam koji, u sadejstvu sa modularnošću, obezbeđuje takav, savršeniji vid upotrebe softvera nosi naziv nasleđivanje. Nasleđivanje je veza između klasa koja podrazumeva preuzimanje sadržaja nekih klasa, odnosno klasa-predaka i na taj način, uz mogućnost modifikacije preuzetog sadržaja i dodavanje novog dobija se klasa-potomak. Klasa (kojih može biti i više) od koje se preuzima sadržaj naziva se klasa-predak, osnovna klasa, klasa davalac ili natklasa. U ovom radu će se koristiti termin natklasa. Klasa koja prima sadržaj naziva se klasa-potomak, izvedena klasa, klasa primalac ili potklasa. U ovom radu će se koristiti termin potklasa. Već je rečeno da se potklasa može izvesti iz više natklasa, međutim, u ovom radu će biti razmatran samo slučaj da je potklasa dobijena od jedne natklase.

Način na koji se razmatra korektnost natklase je već opisan u prethodnom poglavlju. Međutim, da bi se razmatrala korektnost potklase treba uzeti u obzir da je potklasa nastala preuzimanjem nekih sadržaja natklase i dodavanjem novih sadržaja koji su specifični za samu potklasu. Neka je data potklasa A i njena stroga invarijanta I_A i neka je data korektna natklasa B i njena stroga invarijanta I_B. Neka je potklasa A takva da ima samo jednog roditelja i neka je to natklasa B, tada je potklasa korektna ako i samo ako su tačni predikati:

- $\{W_A \wedge Pc_A\}\ c_A()\ \{Qc_A \wedge I_A\}$, za svaki konstruktor $c_A()$ klase A,
- $\{I_A \wedge Pd_A\}\ d_A()\ \{Qd_A \wedge W_A\}$, za svaki destruktor $d_A()$ klase A,
- $\{I_A \wedge Pm_A\}\ m_A()\ \{Qm_A \wedge I_A\}$, za svaki modifikator $m_A()$ klase A,
- $\{I_B \wedge Pm_B\}\ m_B()\ \{Qm_B \wedge I_B\}$, za svaki modifikator $m_B()$ klase B,

pri čemu važi sledeće:

- $W_A \Rightarrow W_B$, gde je $W_A \equiv$"Objekat klase A nalazi se u stanju nepostojanja",
- $(Pc_A \Rightarrow Pc_B) \wedge (Qc_A \Rightarrow Qc_B)$, za svaki konstruktor $c_A()$ klase A,
- svaki konstruktor $c_A()$ klase A u sebi podrazumeva izvršavanje odgovarajućeg konstruktora klase B, odnosno $c_A()\{\ c_B()\ \}$,
- $(Pd_A \Rightarrow Pd_B) \wedge (Qd_A \Rightarrow Qd_B)$, za destruktor $d_A()$ klase A,
- destruktor $d_A()$ klase A u sebi podrazumeva izvršavanje odgovarajućeg destruktora klase B, odnosno $d_A()\{\ d_B()\ \}$,
- $(Pm_A \Rightarrow Pm_B) \wedge (Qm_A \Rightarrow Qm_B)$, za svaki modifikator $m_A()$ klase A, koji menja sadržaj polja koja pripadaju delu specifičnom za potomka,
- $\{Pm_B \wedge I_B\}\ m_B()\ \{Qm_B \wedge I_B\}$, za svaki modifikator $m_B()$ klase B, koji menja sadržaj polja koja pripadaju roditeljskom delu,
- za stroge invarijante važi $I_A \Rightarrow I_B$.

Na osnovu svega što je do sad izloženo, može se doći do zaključka da se stroga invarijanta potklase I_A može prestaviti kao konjunkcija strogih invarijanti natklasa, tj. klasa-predaka I_{B1}, I_{B2},..., I_{Bn} i stroge invarijante dela koji je specifičan za samu potklasu, tj. klasu-naslednicu N. Prema tome, može se pisati $I_A \equiv I_{B1} \wedge I_{B2} \wedge ... \wedge I_{Bn} \wedge N$.

Klasa *PravougliTrougao* se može izvesti iz klase *Trougao*. Već je rečeno da se stroga invarijanta potklase može prestaviti kao konjunkcija strogih invarijanti natklasa i dela koji je specifičan za samu potklasu. To što je specifično za samu klasu *PravougliTrougao* je da je kvadrat nad hipotenuzom jednak zbiru kvadrata nad katetama, pa neka je I_T stroga invarijanta klase *Trougao*, tada je $I_{PT} \equiv I_T \wedge ((a^2=b^2+c^2) \vee (b^2=c^2+a^2) \vee (c^2=a^2+b^2))$ stroga invarijanta klase *PravougliTrougao*.

V. ZAKLJUČAK

Invarijanta klase je svaki predikat koji je tačan u svakom dozvoljenom stanju svakog objekta date klase, pa zbog toga zauzima centralno mesto kada se razmatra korektnost objektno orijentisanog programa. Ako se klasa posmatra kao model klase entiteta, onda je očigledno da pravi smisao invarijante klase jeste da dosledno očuva unutrašnje relacije koje važe između atributa svakog konkretnog entiteta iz date klase entiteta i obezbedi da svaki objekat bude u dozvoljenom stanju, a to znači da bude verodostojan model entiteta, koji može postojati u domenu problema. Odrediti invarijantu klase, a pogotovo strogu invarijantu klase po običaju nije nimalo trivijalno, ali bez obzira na teškoće ovakvo istraživanje predstavlja temelj proizvodnje kvalitetnog softvera i predstavlja veliki izazov za dalji rad i istraživanje metoda i algoritama za određivanje invarijante klase, kao i metoda za njenu primenu u razvoju softverskih sistema.

LITERATURA

[1] B. Meyer, *Object-Oriented Software Construction.* Prentice Hall, 1988.

[2] D. T. Malbaški, *Objekti i objektno programiranje kroz programske jezike C++ i Pascal*. Novi Sad: Fakultet tehničkih nauka, 2006.

[3] M. J. C Gordon, *Programming Language Theory and its implementation*. Prentice Hall, 1988.

[4] E. W. Dijkstra, *A Discipline of Programming*. Prentice Hall, 1976.

[5] D. Cvetković, S. Simić, *Diskretna matematika – matematika za kompjuterske nauke*. Beograd: Naučna knjiga, 1990.

[6] M. Barnett, R. DeLine, M. Fähndrich, K. R. M. Leino, W. Schulte: "Verification of object-oriented programs with invariants," in *Journal of Object Technology*, vol. 3, no. 6, June 2004, Special issue: ECOOP 2003 workshop on FTfJP, pp. 27-56, http://www.jot.fm/issues/issue_2004_06/article2

ABSTRACT

One of the basic criteria for determining software system quality is correctness. Focusing on object oriented programming languages only, primarily on class correctness problem, we arrive at the concept of class invariant. Class invariant is the central point of correctness consideration of object oriented programme and its definition and characteristics are given in this paper. After that, definition of derived class correctness will be given. Some problems have been highlighted and application of class invariant concept have been analysed and in that way a contribution has been given to the research of common principles of quality software system design.

DERIVED CLASS CORRECTNESS

Aleksandar D. Kupusinac, Dušan T. Malbaški