Dodatak

Razlike između verzija jezika UML

Kada se pojavilo prvo izdanje ove knjige, bila je aktuelna verzija 1.0 jezika UML. Izgledalo je da se najveći deo jezika ustalio, i grupa OMG ga je usvajala. Od tada se pojavio veći broj izmena i dopuna. U ovom dodatku opisujem značajne izmene od verzije 1.0 i njihov uticaj na sadržaj ove knjige.

Menjao sam knjigu i usaglašavao je sa standardom UML-a koji je važio u vreme štampanja knjige.

Izmene i dopune jezika UML

Jezik UML pojavio se u verziji 0.8 Objedinjene metode, koja je objavljena na konferenciji OOPSLA u oktobru 1995. godine. To je bio rad Boocha i Rumbaugha, pošto se Jacobson nije priključio kompaniji Rational do tog trenutka. Kompanija Rational je 1996. objavila verzije 0.9 i 0.91, kojima je doprineo i Jacobson. Nakon te poslednje verzije, ime je promenjeno u UML.

Kompanija Rational sa saradnicima podnela je u januaru 1997. verziju 1.0 jezika UML Radnoj grupi za analizu i projektovanje (engl. *Analysis and Design Task Force*, ADTF) organizacije OMG. Zatim su, u septembru 1997. godine, Rational, partneri ove kompanije i drugi predlagači usaglasili svoje radove u jedan predlog OMG standarda za verziju 1.1 jezika UML. Grupa OMG je prihvatila predlog pred kraj 1997. godine. Da bi zabuna bila potpuna, OMG je tom standardu dodelila verziju 1.0. Tako je UML postojao u verziji 1.0 grupe OMG i u verziji 1.1 kompanije Rational. Svi su taj standard nazivali verzijom 1.1.

Od tada je UML poboljšan više puta. Verzija UML 1.2 pojavila se 1998. godine, 1999. godine verzija 1.3, 2001. verzija 1.4, a 2002. verzija 1.5. Većina razlika između verzija 1.x odnosila se na ređe korišćenje detalje, osim verzije

UML 1.3, koja je donela očigledne promene, naročito u slučajevima korišćenja i dijagramima aktivnosti.

Dok je nastajao niz verzija UML 1, tvorci UML-a usredsredili su se na glavne izmene i dopune jezika za verziju UML 2. Prvi Zahtev za predloge (engl. *Request for Proposals*, RFP) izdat je 2000. godine, ali se verzija UML 2 nije ustalila do 2003.

Svakako će se uskoro pojaviti nova poboljšanja UML-a. Više podataka o tome potražite na UML Forumu (**http://uml-forum.com**). Osim toga, neke informacije o UML-u održavam i na svojoj Web lokaciji (**http://martinfowler.com**).

Izmene knjige *UML ukratko*

Kako se pojavljuju nove verzije jezika UML, nastojim da budem u toku kroz izmenjena i dopunjena izdanja knjige *UML ukratko*. Otklanjao sam greške i dopunjavao objašnjenja.

Najdinamičniji period usaglašavanja sa promenama UML-a bio je u vreme nastanka prvog izdanja ove knjige, kada smo često morali da menjamo sadržaj između štampanja tiraža, da bismo bili u toku sa najnovijim standardom jezika. Prvih pet tiraža zasnovani su na verziji UML 1.0. Između pojedinačnih štampanja, UML je malo izmenjen. U šestom tiražu opisana je verzija UML 1.1.

Na verziji UML 1.2 zasnovani su tiraži od sedmog do desetog, a u jedanaestom se prvi put koristi UML 1.3. Na koricama knjiga iz tiraža zasnovanih na verzijama UML-a posle 1.0 odštampan je broj verzije jezika.

Prvih šest tiraža drugog izdanja zasniva se na verziji 1.3. U sedmom tiražu prvi put su uvrštene manje izmene verzije 1.4.

Treće izdanje knjige je izmenjeno prema verziji UML 2 (pogledajte tabelu A.1). U nastavku ovog dodatka sažeto ću opisati glavne promene u UML-u između verzija 1.0 i 1.1, između 1.2 i 1.3 i između 1.x i 2.0. Ne razmatram sve promene, nego samo one koje se odnose na delove knjige *UML ukratko* ili predstavljaju važne karakteristike kojima bih posvetio pažnju.

Nastavljam da sledim osnovne principe ove knjige: razmatram ključne elemente UML-a, koji utiču na primenu jezika u svakodnevnim projektima. Kao i uvek, navodim svoje izbore i savete. Ako postoji bilo kakva neusaglašenost između onoga što napišem i zvanične dokumentacije jezika UML, treba slediti dokumentaciju jezika. (Obavestite me o tome da bih mogao da ispravim greške.)

Tabela A.1 UML ukratko *i odgovarajuće verzije UML-a*

UML ukratko	Verzije UML-a
Prvo izdanje	**UML 1.0-1.3**
Drugo izdanje	**UML 1.3-1.4**
Treće izdanje	**UML 2.0 i dalje**

Iskoristio sam i priliku da ukažem na sve značajne greške i propuste prethodnih tiraža knjige. Zahvaljujem se čitaocima koji su mi skrenuli pažnju na te nedostatke.

Razlike između UML-a 1.0 i 1.1

Tipovi i klase realizacije

U prvom izdanju knjige *UML ukratko* pisao sam o različitim gledištima i kako su ona promenila način crtanja i tumačenja modela, posebno dijagrama klasa. U jeziku UML sada je to uzeto u obzir, pa se kaže da se sve klase na dijagramu mogu svrstati ili u tipove ili u klase realizacije.

Klasa realizacije (engl. *implementation class*) odgovara klasi u razvojnom okruženju. **Tip** (engl. *type*) nije toliko jasan pojam, pošto predstavlja apstrakciju koja nije povezana sa realizacijom. To bi mogao biti CORBA tip, klasa sa stanovišta specifikacije ili klasa sa konceptualnog stanovišta. Ako je neophodno, možete dodati i stereotipove radi preciznije podele.

Možete označiti da sve klase nekog dijagrama slede određen stereotip. To činite kada dijagrame crtate prema određenom stanovištu. Sa stanovišta realizacije koristili biste klase realizacije, dok biste sa stanovišta specifikacije ili koncepata koristili tipove.

Koristite vezu realizacije da biste ukazali na to da klasa realizuje jedan ili više tipova.

Postoji razlika između tipa i interfejsa. Interfejs treba neposredno da odgovara interfejsu jezika Java ili modela COM. Zato interfejsi imaju samo operacije, ali ne i atribute.

U slučaju klasa realizacije možete primeniti samo jednostruku, statičku klasifikaciju, dok su u slučaju tipova moguće višestruka i dinamička klasifikacija.

(Pretpostavljam da je razlog to što se u glavnim objektno orijentisanim jezicima primenjuje jednostruka, statička klasifikacija. To ograničenje ne bi trebalo da važi ako nekada budete koristili jezik koji podržava višestruku ili dinamičku klasifikaciju.)

Oznaka complete i nepotpunost u razlikovanju ograničenja

U ranijim verzijama knjige *UML ukratko*, napisao sam da ograničenje generalizacije {complete} ukazuje na to da svaka instanca nadtipa mora biti instanca podtipa unutar te particije. Po verziji UML 1.1, {complete} znači da su svi podtipovi unutar particije definisani, što ima drugi smisao. Pronašao sam neke nedoslednosti u tumačenju ovog ograničenja, pa budite oprezni. Ako hoćete da ukažete na to da svaka instanca nadtipa treba da bude instanca jednog od podtipova, preporučujem vam da koristite drugo ograničenje, kako ne bi došlo do zabune. Trenutno za to koristim ograničenje {mandatory}.

Kompozicija

U verziji UML 1.0 podrazumevalo se da je veza kompozicije bila nepromenljiva, ili zamrznuta, bar u slučaju komponenata sa jednom vrednošću. To ograničenje više nije propisano standardom.

Nepromenljivost i oznaka frozen

U UML-u je definisano ograničenje {frozen}, koje pokazuje nepromenljivost uloga u asocijaciji. Po važećoj definiciji, izgleda da se ograničenje ne primenjuje na atribute, niti na klase. Pri projektovanju koristim pojam **zamrznut** (engl. *frozen*) umesto nepromenljivosti i rado primenjujem to ograničenje na uloge u asocijaciji, klase i atribute.

Povratak na dijagramima sekvence

U verziji UML 1.0 povratak na dijagramima sekvence označavao se prelomljenim vrhom strelice a ne punim vrhom. Taj način crtanja bio je problematičan, pošto je razlika bila neznatna i lako ju je bilo prevideti. U verziji UML 1.1 koristi se isprekidana strelica za označavanje povratka, što mi se više dopada, pošto se tako povratak iz funkcije mnogo lakše uočava. (Koristio sam isprekidane linije povratka u knjizi *Analysis Patterns* [Fowler, AP].) Radi kasnije upotrebe rezultata možete mu dodeliti ime, na primer `dovoljnoZaliha := proveri()`.

Upotreba pojma „uloga“

U verziji UML 1.0 izraz **uloga** (engl. *role*) uglavnom je ukazivao na smer asocijacije. U verziji UML 1.1 ovakav način upotrebe naziva se **uloga u asocijaciji** (engl. *association role*). Postoji i **uloga u kolaboraciji** (engl. *collaboration role*), što predstavlja ulogu koju instanca ili klasa imaju u kolaboraciji. Verzija UML 1.1 se mnogo više usredsređuje na kolaboraciju, pa izgleda da bi to mogla postati osnovna primena pojma „uloga“.

Izmene od verzije UML 1.2 (i 1.1) do verzije 1.3 (i 1.5)

Slučajevi korišćenja

U verziji UML 1.1 postojale su dve vrste veza, «uses» i «extends», a obe su stereotipovi generalizacije. Verzija UML 1.3 nudi tri veze:

- Mehanizam «include» je stereotip zavisnosti. On ukazuje na to da je jedan slučaj korišćenja uključen u drugi. Obično se to događa kada nekoliko slučajeva korišćenja imaju zajedničke korake. Pomoću uključenog slučaja korišćenja možete izdvojiti zajedničko ponašanje. U primeru bankomata, slučajevi korišćenja Isplati novac i Prenesi novac koriste slučaj Proveri klijenta. Ta zavisnost zamenjuje uobičajenu upotrebu rezervisane reči «uses».
- **Generalizacija** (engl. *generalization*) slučaja korišćenja ukazuje na to da je jedan slučaj korišćenja varijacija drugog. Na taj način bismo mogli prikazati jedan, osnovni, slučaj korišćenja Isplati novac i poseban slučaj korišćenja, za obradu situacije kada se isplata odbija usled nedovoljne količine novca na računu. Odbijanje isplate bi se moglo obraditi kao specijalan slučaj korišćenja za isplatu. (Možete rešiti odbijanje isplate i kao poseban scenario unutar slučaja korišćenja Isplati novac.) Takav specijalan slučaj korišćenja može promeniti bilo koji deo osnovnog slučaja korišćenja.
- Mehanizam «extend» je stereotip zavisnosti – oblik proširenja kojim se lakše upravlja nego generalizacijom. Osnovni slučaj korišćenja deklariše tačke proširenja. Slučaj korišćenja, koji predstavlja proširenje, može menjati ponašanje samo u tim tačkama proširenja. Na primer, u slučaju korišćenja koji opisuje kupovinu proizvoda putem Interneta, deklarišu se tačke proširenja za prikupljanje podataka o isporuci i za prikupljanje podataka o načinu plaćanja. Taj slučaj korišćenja mogao bi se proširiti za redovnog kupca, o kome se podaci mogu dobiti na neki drugi način.

Postoji izvesna zabuna oko odnosa između starih i novih tipova veza. Većina projektanata koristila je rezervisanu reč «uses» kao što je u verziji 1.3 korišćena rezervisana reč «includes», tako da «includes» uglavnom zamenjuje «uses». Takođe, većina je koristila «extends» iz verzije 1.1 i na način koji omogućava bolje upravljanje u verziji 1.3 i kao opšti mehanizam za redefinisanje u generalizaciji verzije 1.3. Zato možete pomisliti da je «extends» iz verzije 1.1 podeljen na «extends» verzije 1.3 i generalizaciju.

Iako ovo objašnjenje obuhvata najveći broj UML dijagrama koje sam video, to nije sasvim ispravan način korišćenja starih veza. Međutim, većina projektanata nije strogo sledila pravila, pa ni ja ne nameravam da se u to upuštam.

Dijagrami aktivnosti

Kada je jezik UML došao do verzije 1.2, javila su se pitanja o značenju dijagrama aktivnosti. Zato je u verziji 1.3 posvećeno dosta pažnje strožem definisanju semantike tih dijagrama.

U slučaju uslovnog ponašanja, i grananje i stapanje aktivnosti odlučivanja predstavlja se oznakom romba. Iako ni oznake završetka grananja, ni oznake stapanja nisu neophodne za opisivanje uslovnog ponašanja, sve više se koriste, tako da se uslovno ponašanje može vizuelno izdvojiti.

Oznaka za sinhronizaciju se sada naziva **grananje** (engl. *fork*) u slučaju deljenja upravljanja, ili **spajanje** (engl. *join*) pri sinhronizaciji upravljanja. Međutim, spajanju više ne možete dodavati proizvoljne uslove. Osim toga, morate slediti pravila koja obezbeđuju slaganje grananja i spajanja. To znači da za svako grananje mora postojati odgovarajuće spajanje, koje okuplja niti aktivirane grananjem. Možete ugnežđivati grananja i spajanja, a možete ih i ukloniti sa dijagrama ako su niti nacrtane između dva grananja ili između dva spajanja.

Spajanja se aktiviraju tek po izvršenju svih ulaznih niti. Međutim, na izlaznu nit grananja možete dodati uslov. Ako taj uslov nije ispunjen, odgovarajuća nit se smatra izvršenom sa stanovišta spajanja.

Višestruki okidači više ne postoje. Umesto njih postoje dinamičke paralelnosti u okviru aktivnosti, označene zvezdicom (*). Takva aktivnost se može više puta paralelno pozvati, a sva izvršenja se moraju okončati pre početka bilo kog izlaznog prelaza. Ova mogućnost je slična višestrukom okidaču i odgovarajućem uslovu sinhronizacije, mada je manje prilagodljiva.

Navedena pravila smanjuju prilagodljivost dijagrama aktivnosti, i u potpunosti ih čine specijalnim slučajevima dijagrama mašina stanja. Veza između dijagrama aktivnosti i mašina stanja bila je predmet rasprave u grupi za revizije (engl. *Revision Task Force*, RTF). U sledećim verzijama jezika UML može se potpuno promeniti izgled dijagrama aktivnosti.

Izmene od verzije 1.3 do verzije 1.4

U verziji UML 1.4 najuočljivija promena je dodavanje **profila** (engl. *profiles*), što omogućava grupisanje proširenja u koherentan skup. Dokumentacija jezika UML navodi i nekoliko primera profila. Uz to, definicija stereotipa postaje formalnija, a elementi modela mogu imati više stereotipova (u verziji UML 1.3 važilo je ograničenje da može postojati jedan stereotip).

Jeziku UML su dodati **artefakti** (engl. *artifacts*). Artefakt je fizička manifestacija neke komponente – na primer, biblioteka Xerces je komponenta, a kopije na mom disku su artefakti koji realizuju tu komponentu.

Pre verzije 1.3, u metamodelu UML-a nije se moglo rešiti pitanje **paketne vidljivosti** (engl. *package visibility*) jezika Java. Sada se za to koristi simbol „~".

U verziji UML 1.4 prelomljen je vrh strelice na dijagramima interakcije korišćen za označavanje asinhronosti, što je zbunjujuća promena, nekompatibilna sa prethodnim verzijama. Neke je to neprijatno iznenadilo, uključujući i mene.

Izmene od verzije 1.4 do verzije 1.5

Osnovna izmena se sastojala u dodavanju semantike akcija, što je bio neophodan korak da bi UML postao programski jezik. Izmena je uvedena ranije da bi bila dostupna pre izdavanja potpune verzije UML 2.

Od jezika UML 1.x do UML 2.0

U verziji UML 2 jezik je najviše promenjen. Promenjene su razne vrste elemenata jezika, a mnoge promene su se odrazile na knjigu *UML ukratko*.

Unutar jezika je suštinski promenjen metamodel UML-a. Iako te promene ne utiču na sadržaj ove knjige, one su nekima veoma značajne.

Jedna od očiglednih promena je uvođenje novih tipova dijagrama. Dijagrami objekata i dijagrami paketa i ranije su često korišćeni, iako su tek u verziji 2.0 postali standardni tipovi dijagrama. Dijagrami kolaboracije su promenili ime, i u verziji UML 2 zovu se dijagrami komunikacije. Uvedeni su novi dijagrami pregleda interakcije, vremenski dijagrami i dijagrami složene strukture.

Veliki broj promena nije uticao na *UML ukratko*. Izostavio sam razmatranje proširenja mašina stanja, kapija u dijagramima interakcije i jakih tipova u dijagramima klasa.

U nastavku razmatram samo promene koje su uticale na sadržaj knjige *UML ukratko*. Radi se o promenama elemenata koje sam opisao u prethodnim izdanjima ili novim elementima, prvi put opisanim u ovom izdanju. Pošto su promene brojne, organizovao sam ih prema poglavljima.

Dijagrami klasa: osnovni pojmovi (poglavlje 3)

Atributi i jednosmerne asocijacije sada su, pre svega, različiti načini označavanja istog pojma svojstva. Izostavljene su prekidne kardinalnosti, na primer [2, 4]. Izostavljeno je zamrznuto svojstvo. Dodao sam niz opštih rezervisanih reči za zavisnosti, od kojih su neke nove u verziji UML 2. Rezervisane reči «parameter» i «local» izostavljene su.

Dijagrami sekvence (poglavlje 4)

Velika promena je uvođenje oznake okvira interakcije u dijagramima sekvence, radi prikazivanja iterativnih, uslovnih i raznih drugih oblika upravljanja ponašanjem. To omogućava da dijagramima sekvence potpuno opišete algoritme, mada nisam ubeđen da su ti dijagrami jasniji nego kôd. Iz dijagrama sekvence su izostavljene stare oznake iteracija i uslovi na porukama. Na početku linije života više se ne nalaze instance, nego na njih ukazujem koristeći izraz **učesnik** (engl. *participant*). Dijagrami kolaboracije iz verzije UML 1 sada se pojavljuju pod novim imenom – dijagrami komunikacije.

Dijagrami klasa: napredni elementi (poglavlje 5)

Sada su stereotipovi mnogo čvršće povezani sa jezikom. Posledica toga je da reči između oznaka « i » nazivam rezervisanim rečima, a neke od njih su stereotipovi. Instance na dijagramima objekata sada su specifikacije instanci. Klase mogu zahtevati ili realizovati interfejse. Višestruka klasifikacija koristi generalizacione skupove da bi grupisala generalizacije. Za crtanje komponenata više se ne koristi poseban simbol. Oznake aktivnih objekata sadrže dvostruke uspravne linije umesto podebljanih linija.

Dijagrami mašine stanja (poglavlje 10)

U verziji UML 1 postojala je razlika između kraćih akcija i dužih aktivnosti. U verziji UML 2 koristi se jedinstven pojam aktivnost, a dugotrajne aktivnosti su nazvane izvršne aktivnosti.

Dijagrami aktivnosti (poglavlje 11)

U verziji UML 1, dijagrami aktivnosti bili su specijalan slučaj dijagrama stanja. U verziji UML 2 ta veza je prekinuta i zato su ukinuta pravila usklađivanja grananja i spajanja, koja su se morala poštovati u dijagramima stanja te verzije. Posledica je da se dijagrami aktivnosti mogu bolje razumeti kada se opisuje tok žetona nego prelaz stanja. Pojavio se niz novih oznaka, uključujući vremenske signale i prijem signala, parametre, specifikacije spajanja, nožice, transformacije toka, račve, oblasti primene i završetke toka.

Jednostavna, mada zbunjujuća, izmena je da se u verziji UML 1 podrazumevalo stapanje više ulaznih tokova aktivnosti, dok se u verziji UML 2 podrazumeva spajanje. Iz tog razloga preporučujem da na dijagramima aktivnosti izričito zadate stapanje ili spajanje.

Plivačke staze sada mogu biti i višedimenzionalne i zovu se particije.