UML potřetí a naposledy

Dosud jste se v našem miniseriálu seznámili s historií unifikovaného modelovacího jazyka UML a s jeho nejčastěji používanými diagramy. Na závěr doplníme některé další druhy diagramů, podíváme se na celkovou strukturu UML a napovíme, kde získat další informace.

Diagram stavů

Dalším druhem diagramů, které popisují různé aspekty chování modelovaného systému, jsou diagramy stavů (stavové diagramy). Tyto diagramy popisují změny stavu celého systému nebo vybraného objektu v průběhu času. Vyjadřují posloupnosti stavů, kterými může modelovaný systém procházet, jejich vzájemnou návaznost, vlastnosti těchto stavů, ale také události, akce a další okolnosti, které změny stavů modelovaného systému doprovázejí. (Někdy se místo o "stavech" hovoří také o "režimech".)

Diagram stavů začíná plným černým kroužkem a končí terčíkem (černým kroužkem uvnitř bílého). Stav systému znázorňujeme ikonou, která má tvar obdélníka se zaoblenými rohy, v němž je zapsáno jméno stavu a případně další informace. Přechody mezi stavy vyjadřujeme šipkou směřující od předcházejícího k následujícímu stavu; k této šipce můžeme připsat událost, která změnu stavu způsobí (tzv. spouštěcí událost), nebo podmínku, po jejímž splnění dojde ke změně stavu (tzv. strážní podmínka, anglicky guard).

Jako (velice zjednodušený) příklad si na obrázku 1 ukážeme stavový diagram práce počítače vybaveného operačním systémem s grafickým uživatelským rozhraním (převzato z [4]).

Po zapnutí přejde počítač do stavu Inicializace; po jeho dokončení přejde do stavu Běh. Příkaz k vypnutí, zadaný prostřednictvím uživatelského rozhraní, způsobí přechod do stavu Ukončování, po němž činnost počítače skončí. Ze stavu Běh může počítač přejít do stavu Úsporný režim, jestliže uplyne předepsaný časový interval nečinnosti (rozumí se nečinnosti uživatele); to je příklad výše zmíněné strážní podmínky. Z Úsporného režimu může počítač přejít jedině zpět do stavu Běh.

Obrázek obsahuje většinou jen zjednodušené ikony stavů. Pouze pro stav Inicializace jsme použili úplnou ikonu. Ta je rozdělena dvěma vodorovnými čarami na tři části; v horní je uvedeno jméno stavu, ve střední stavové proměnné a jejich hodnoty a ve spodní akce, které je třeba po přechodu do tohoto stavu provést. Akce je uvedena návěštím proveď/ (anglicky do/), za kterým následuje její popis. Mohou tu být zapsány i vstupní a výstupní operace, které jsou uvedeny návěštím vstup/, resp. výstup/ (anglicky input/, resp. output/).

Podstavy a paralelní stavy

Při podrobnějším zkoumání se často ukáže, že některé stavy lze rozložit na několik "jemnějších" stavů; pak hovoříme o podstavech a nadřízených stavech. Podstavy a jejich vztahy zakreslujeme do ikony nadřízeného stavu.

Občas také zjistíme, že některé podstavy nastávají současně; ty označujeme jako paralelní (souběžné) a zakreslujeme je do ikony nadřízeného stavu oddělené vodorovnou přerušovanou čarou.

Vraťme se k předchozímu příkladu. Model popsaný stavovým diagramem z obr. 1 sice vystihuje skutečnost, při bližším pohledu ale zjistíme, že stav Běh se vlastně skládá z několika podstavů, které nazveme Čekání na vstup od uživatele, Zpracování vstupu a Zobrazení výsledků vstupu. Vedle toho většina operačních systémů s grafickým uživatelským rozhraním zobrazuje čas. To znamená, že musí sledovat systémové hodiny a po uplynutí jistého intervalu - např. jedné sekundy - aktualizovat zobrazený údaj. Z toho plyne, že se zde zároveň střídají stavy Sledování systémového času a Aktualizace zobrazeného údaje. Upravenou ikonu stavu Běh ukazuje obrázek 2.

Některé stavy je nutné při přechodu do následujícího stavu uložit. To se týká např. stavu Běh v předchozím příkladu při přechodu do úsporného režimu. Uživatel, který odběhne od počítače uprostřed rozdělané práce a zdrží se, nesmí tím, že jeho počítač přejde do Úsporného režimu, ztratit, co předtím udělal. I to lze pochopitelně v UML vyznačit.

Diagram sekvencí

Tento diagram popisuje vzájemnou interakci objektů systému v závislosti na čase. V diagramu jsou zachyceny kromě jednotlivých objektů také zprávy, které si tyto objekty posílají. Přitom rozlišujeme jednoduché, synchronní a asynchronní zprávy. Dále si povíme, co to znamená.

Diagram sekvencí začíná zjednodušenými ikonami objektů, jejichž interakce sledujeme; typicky jsou v řadě vedle sebe. Od každé z ikon objektů vede svislá přerušovaná čára, které říkáme čára života (lifeline). Na ní pomocí obdélníků vyznačujeme období, kdy je objekt aktivní. Zprávu, kterou jeden objekt pošle jinému, vyznačíme vodorovnou šipkou, která směřuje od odesílatele k příjemci. Přitom platí:

* Jednoduché zprávy, které znamenají přenos řízení, vyznačujeme obyčejnou šipkou.

* Synchronní zprávy, u nichž odesílatel čeká na odpověď a do té doby nic nedělá, označujeme šipkou zakončenou vyplněným trojúhelníkem. (Odesláním synchronní zprávy spustíme metodu přijímajícího objektu a teprve po jejím ukončení se pokračuje dál; jde o analogii spuštění podprogramu ve strukturovaném programování.)

* Asynchronní zprávy, tj. zprávy, u kterých odesílatel nečeká na odpověď, označujeme šipkou s polovinou hrotu. (Odesláním asynchronní zprávy pouze spustíme metodu přijímajícího objektu, nečekáme na výsledek a pokračujeme dál.)

* Návrat (return) označujeme obyčejnou šipkou s čárkovanou čarou, ne vždy ho však v diagramech sekvencí uvádíme.

Zrušení objektu vyjádříme tím, že jeho čáru života ukončíme křížkem. Čas je v tomto diagramu naznačen polohou ve svislém směru; čím dále jsme od horního okraje, tím více času od počátku uplynulo.

Diagram sekvencí umožňuje také vyjádřit opakované zprávy, podmíněné zprávy, parametry zpráv, vysílané zprávy (tj. zprávy určené všem objektům, nikoli jen určitému příjemci) atd.

Svislé obdélníkové úseky na čarách života objektů představují vlastně běžící metody. Pokud objekt v čase postupně provádí více metod, zakreslíme do diagramu více úseků pod sebe. Čáry mezi úseky lze chápat jako stavy objektu (mezi jeho aktivitami). Tento diagram dovoluje zakreslit i cykly a větvení, takže vhodně doplňuje stavové diagramy.

Poznamenejme, že zatímco stavové diagramy představují detailní popis aktivit objektů v čase, diagramy sekvencí nabízejí souhrnný pohled na časové návaznosti aktivit jednotlivých objektů propojených spolu do téhož systému.

Jako příklad poslouží silně zjednodušené schéma práce s klávesnicí v operačním systému DOS na obrázku 3.

Nejprve uživatel stiskne klávesu. (Uživatel PC není součástí modelovaného systému, proto je zde znázorněn ikonou aktéra.) Tím pošle klávesnici zprávu (asynchronní, neboť zpravidla nečeká na výsledek a píše dál). Klávesnice pošle zprávu obsahující polohu stisknuté klávesy ovladači klávesnice. Tentokrát jde o synchronní zprávu, neboť klávesnice čeká, dokud nedostane informaci, že byla předchozí zpráva zpracována.

Ovladač klávesnice znak zpracuje (podle platné kódovací tabulky zjistí, jaký znak tato zpráva reprezentuje) a pošle klávesnici zprávu, že znak byl zpracován; tím jí umožní odesílat další znaky. Zároveň ovladač klávesnice odešle asynchronní zprávu obsahující tento znak kruhové frontě, která slouží jako sklad přijatých znaků.

Kruhová fronta se pokusí tento znak uložit. Pokud se to nepodaří, neboť úložný prostor je již vyčerpán, odešle zprávu zvukovému zařízení, které upozorní uživatele, že píše příliš rychle nebo že znaky žádný program neodebírá.

Diagram činností

Posledním diagramem, o kterém si zde povíme, bude diagram činností (aktivit). Na první pohled to není nic jiného než starý známý vývojový diagram (kdysi také důvěrně zvaný "blokáč"), pohřbený s velkou slávou v sedmdesátých letech při boji za strukturované programování, nyní nenápadně vzkříšený a zabalený v trochu jiném (a mocnějším) hávu. Skutečnost je ale složitější. Vedle jasného vlivu vývojových diagramů zde najdeme prvky Petriho sítí, diagramů událostí J. Odella a dalších objektových i neobjektových modelovacích technik.

Diagram činností znázorňuje algoritmy. Ukazuje pořadí kroků, větvení procesu atd. Začíná a končí stejnými značkami jako diagram stavů; činnosti se v něm vyjadřují obdélníkem se dvěma polokruhovými konci, v němž je název nebo popis činnosti. Spojnice mezi těmito ikonami představují šipky směřující od činnosti právě ukončené k činnosti následující. Rozhodování se znázorňuje např. čtvercem postaveným na jeden z vrcholů, z něhož vycházejí spojnice odpovídající jednotlivým alternativám; ke každé spojnici je připsána podmínka, která danou alternativu určuje. (V UML máme i jiné možnosti, jak znázornit rozhodování.)

Jako jednoduchý příklad si na obrázku 4 ukážeme diagram činností popisující výpočet faktoriálu nezáporného čísla n. Na počátku vložíme do pomocné proměnné s hodnotu 1. Pak zkontrolujeme, je-li n > 1, a pokud ano, uložíme do s hodnotu s*n, zmenšíme n o 1 a vrátíme se na rozhodování, zda je n > 1. Jestliže už tato podmínka splněna není, obsahuje proměnná s výsledek, a proto skončíme.

Diagramy činností mohou ovšem zachytit i další skutečnosti. Jestliže se na popisované činnosti podílí několik objektů, rozdělíme diagram svislými čárkovanými čarami na tzv. paralelní dráhy (anglicky

označované swimlines podle podobnosti s plaveckou dráhou v závodním bazénu). Do záhlaví každé z paralelních drah zapíšeme jméno jednoho z objektů; činnosti, které tento objekt vykonává, pak zakreslujeme do ní.

Chceme-li vyjádřit, že se modelovaný proces rozdělí na několik paralelně probíhajících podprocesů, nakreslíme silnou vodorovnou úsečku, do níž vstupuje jedna spojnice a z níž vystupují spojnice odpovídající jednotlivým podprocesům. Podobně vyjádříme spojení několika podprocesů.

Diagramy činností umožňují také zachytit odeslání a přijetí zprávy.

O čem jsme nehovořili

Na závěr povídání o diagramech používaných v UML si alespoň naznačíme, co jsme vynechali. Především jsme nehovořili o diagramech komponent, které vyjadřují seskupení tříd do softwarových komponent a jejich vzájemné vztahy. Nehovořili jsme také o diagramech spolupráce (collaboration diagrams), které jsou v podstatě ekvivalentní diagramům sekvencí, mohou ale zdůraznit trochu jiný pohled na modelovaný systém. (Tento diagram ponechává stranou časovou následnost zpráv, ale zato umožňuje zakreslit objekty vedle sebe v celé ploše a zdůraznit tak souvislost mezi zprávami, které si objekty posílají, a asociacemi mezi nimi.) Nezmínili jsme se ani o diagramu nasazení (deployment diagram), který vyjadřuje rozmístění softwarových komponent v hardwarových součástech výsledného systému.

Stranou jsme ponechali i skutečnost, že jazyk UML se nebrání zavádění nových druhů diagramů, pokud zjistíme, že je potřebujeme. Můžete dokonce používat smíšené diagramy, tj. diagramy, které obsahují prvky z různých druhů diagramů. Nic nám např. nebrání používat v diagramech sekvencí ikony stavů a zachycovat tak vývoj stavů součástí systému v čase, vnořit diagram činností do ikony třídy a zachytit tak algoritmus implementovaný metodou této třídy atd.

Vynechali jsme možnosti parametrizace tříd a jiných konstrukcí vzhledem k datovým typům či jiným údajům, využití návrhových vzorů atd. Vedle toho musíme zdůraznit, že výklad v našem miniseriálu byl zjednodušený, že jsme vynechali řadu méně často používaných možností, klíčových slov, alternativních způsobů zobrazení v diagramu a dalších informací.

Model, metamodel a tak dál

Podívejme se nyní trochu blíže na strukturu unifikovaného modelovacího jazyka. Pokud jste získali dojem, že jde prostě o jakousi skupinu ikon a pravidel, jak je spojovat do nějakých diagramů, pak sice nejste úplně vedle, ale nemáte také úplnou pravdu. Zkusme se tedy podívat na UML "z nadhledu". Učebnice UML říkají, že tento jazyk má čtyři vrstvy, které se zpravidla znázorňují podobně jako na obrázku 5.

Nejnižší vrstva, vrstva uživatelských objektů, představuje konkrétní softwarové objekty z konkrétního problému, které se ve výsledné aplikaci objeví. ("Uživatel" zde znamená uživatele jazyka UML, nikoli uživatele výsledné aplikace.) V této vrstvě se pohybujeme zpravidla při rozhovorech se zadavateli úlohy a s potenciálními uživateli výsledného systému, když shromažďujeme data potřebná pro vytvoření modelu.

Model je, jak víme, představován skupinou diagramů v UML. Ve druhé vrstvě se tedy pohybujeme při analýze, kdy již pracujeme s třídami, s případy užití a s dalšími aspekty řešeného problému, zobrazujeme je pomocí odpovídajících diagramů a tak vytváříme model.

Ve třetí vrstvě se pohybujeme, když se například učíme UML. Náš výklad v tomto článku - kromě příkladů - probíhal právě ve vrstvě metamodelu, neboť vysvětloval smysl, pravidla vytváření a použití jednotlivých diagramů jazyka UML pro konstrukci požadovaného modelu. (Název metamodel pochází z toho, že tato vrstva vlastně představuje model modelovacího procesu.)

Zpravidla se říká, že vrstva metamodelu v UML obsahuje základ jazyka UML, prvky chování a nástroje pro správu modelu. Do základu UML patří mj. datové typy používané v UML, mechanismy pro rozšíření jazyka, jako jsou stereotypy, a další součásti. Mezi prvky chování najdeme nástroje pro zachycení dynamických prvků modelu - např. případy užití, popisy stavů atd. Poznamenejme, že v této vrstvě musí uvažovat nejen ti, kdo UML učí, ale např. i ti, kdo navrhují editory jazyka UML a vývojové nástroje CASE, které ho budou používat.

Nejobecnější vrstvou je vrstva metametamodelu; můžeme si ji představovat jako celkový model unifikovaného modelovacího jazyka UML. V této vrstvě se pohybují ti, kdo pracují na vývoji UML.

Zdroje informací

Na závěr se podívejme, kde se lze o jazyku UML dozvědět více. Asi nejvýznamnějším zdrojem informací o něm jsou knihy autorů jazyka UML [1] a [2], které je možné považovat za základní literaturu o tomto jazyku. Jako učebnice se nám ovšem nejvíce líbila kniha M. Fowlera [3], která je poměrně stručná a dobře se čte.

Předchozí publikace jsou zatím k dispozici pouze v angličtině. Pokud jde o publikace v češtině, na našem trhu je k disposici kniha [4] s výmluvným názvem Myslíme v UML, která se v originále jmenovala Teach yourself UML in 24 hours. Najdeme v ní poměrně vyčerpávající výklad o všech diagramech a rozsáhlý příklad, který ukazuje použití UML při analýze činnosti a návrhu softwaru pro fiktivní nadnárodní restaurační firmu; tento příklad zahrnuje i modelování vestavěných systémů. Česká terminologie použitá v této knize byla konzultována s odborníky používajícími UML nejen ve firemní praxi, ale i ve vysokoškolské výuce.

Nedávno také vyšla kniha [5], která používá UML jako nástroj pro výklad o objektově orientovaném návrhu. První polovina knihy je proto zasvěcena výkladu tohoto jazyka, druhá pak problematice objektově orientovaného návrhu a jeho častým chybám. Poznamenejme ale, že česká terminologie použitá v tomto překladu se liší od [4] a je v některých místech značně "svévolná". (Přitom obě knihy vyšly ve stejném nakladatelství!)

Řadu příkladů modelů založených na UML najdeme v knize [6]; jsou sice poněkud jednostranně zaměřeny na "datařinu", ale to příliš nevadí, neboť právě ta, jak známo, dnes hýbe světem.

Vojtěch Merunka, Miroslav Virius

Literatura

[1] Grady Booch, James Rumbaugh, Ivar Jacobson: The Unified Modeling Language User Guide. Addison-Wesley, 1999.

[2] Grady Booch, James Rumbaugh, Ivar Jacobson: The Unified Modeling Language Reference Manual. Addison-Wesley, 1999.

[3] Martin Fowler, Kendall Scot: UML Distilled. 2nd Edition. Addison-Wesley, 2000.

[4] Joseph Schmuller: Myslíme v jazyku UML. Grada Publishing, 2001.

[5] Meilir Page-Jones: Základy objektově orientovaného návrhu v UML. Grada Publishing, Praha 2001.

[6] Ľubomír Šešera: Datové modelování v příkladech. Grada Publishing, Praha 2001.