QUO VADIS PROGRAMOVÁNÍ?

Rudolf Pecinovský
ICZ a.s., Na hřebenech II 1718/10, 147 00 Praha 4,
VŠE Praha, Fakulta informatiky a statistiky, Katedra informačních technologií
rudolf@pecinovsky.cz

ABSTRAKT:

Příspěvek shrnuje vývoj programovacích paradigmat v posledních desetiletích a na ně navazující vývoj vyučovaných paradigmat a metodik jejich výuky. Poté rozebírá některé současné trendy v programování a těžkopádnou reakci výukových programů na tyto trendy. Poukazuje na stále sílící pozici nejrůznějších frameworků a na přetrvávající ignoraci tohoto trendu v učebních plánech škol všech stupňů.

KLÍČOVÁ SLOVA:

Výuka programování, framework, učební plán.

1 ÚVOD

Pravidelnou součástí této konference jsou diskuse o tom, jak učit programování tak, aby tato výuka vybavila studenty optimálně pro praxi. V dávnějších letech se vedly diskuse o tom, zda učit nebo neučit (případně prosazovat či neprosazovat) strukturované programování (viz např. [12]). Později pak začalo strukturované programování soupeřit s nově se prosazujícím programováním objektově orientovaným ([1], [2]). To, že se však nějaké programovací paradigma prosadilo v praxi, však ještě vůbec neznamená, že se prosadilo také ve výuce. Po přednáškách propagujících používání objektově orientovaného programování proto nastoupily přednášky propagující jeho zařazení do výuky.

2 PROBLÉMY S VÝUKOU A NASAZENÍM OOP

Vzápětí po prvních úvahách o tom, že by bylo vhodné zařadit výklad OOP do výuky programování, začala vyvstávat i otázka JAK objektově orientované programování učit. První kurzy učily OOP většinou tak, že nejprve probraly všechny témata týkající se strukturovaného programování a poté přidaly výklad objektových konstrukcí jako takovou zajímavou nadstavbu umožňující zvýšit produktivitu práce programátora.

Tento přístup se záhy ukázal jako poměrně nevhodný, protože vůbec nerespektoval to, že objektově orientované programování přináší zcela nové myšlenkové paradigma, které je paradigmatu strukturovaného programování velmi vzdáleno. Studenti, kteří se naučili v první fázi kurzu programovat strukturovaně, brali objektové konstrukce pouze jako zajímavou nadstavbu, která jim v některých situacích ulehčí práci, ale způsob svého uvažování nijak nezměnili, takže jejich programy byly nadále strukturované, byť občas používaly objektové konstrukce. Skutečných výhod, které nabízí OOP, proto v žádném případě nedosáhli a v mnoha případech ještě špatným použitím objektových konstrukcí zanesli do architektury destabilizační prvky. Řada z těchto programátorů nakonec obvinila ze špatně funkce programu právě použití OOP a na toto paradigma zanevřela.

3 NOVÉ METODIKY VÝUKY

Na tuto skutečnost reagovaly nové metodiky výuky programování, které se snaží učit programování tak, aby studenti doopravdy pochopili podstatu objektově orientovaného paradigmatu a správě je ve svých projektech použili. Kolem roku 2000 vznikla metodika *Object First*, která prosazovala zahájení výuky přímo výkladem objektů a prací s nimi. Klasické konstrukce

strukturovaného programování (podmínky, cykly, …) doporučovala vykládat až poté, co si studenti osvojili základy práce s objekty bez použití těchto konstrukcí.

Metodika *Object First* se však soustředila pouze na objekty jako základní konstrukce, avšak konstrukci rozhraní nechávala až na konec základního výkladu a o návrhových vzorech se v základním kurzu prakticky nezmínila. Rozhraní a návrhové vzory se však mezi tím staly klíčovými složkami návrhu objektově orientovaných programů a jejich opožděné zařazení neumožňovalo studentům poznat a pochopit jejich skutečný význam v moderním programování.

V roce 2004 byla proto představena metodika *Design Patterns First*, která seznamuje studenty s rozhraním hned na počátku výkladu a prakticky od počátku kurzu také zařazuje témata přibližující použití (a tím i význam) návrhových vzorů. Tato metodika navíc doporučuje upravit výklad některých základních programových konstrukcí tak, aby je studenti lépe pochopili. Mezi konstrukce, jejichž výklad oproti dosavadním zvyklostem zpřesňuje či dokonce upravuje, patří:

- vysvětlení pojmu objekt,
- rozdílné chápání tříd a objektů,
- rozhraní,
- konstrukce objektů a konstruktory,
- klíčové slovo `this`,
- dědičnost tříd.

Podrobný výklad všech úprav lze nalézt např. v [3], [4] nebo [5].

4 DALŠÍ RYSY PROGRAMOVÁNÍ

Již delší dobu si zákazníci uvědomují, že cena vývoje první verze programu je pouze zlomkem ceny, kterou bude třeba zaplatit v průběhu celého života programu. Stále častěji proto požadují vytvoření programů, které by minimalizovaly celkové náklady, které je na ně třeba vynaložit po dobu jejich života. (Tyto otázky jsou podrobněji probírány např. v [6].)

I na to bychom měli ve výuce reagovat a učit studenty nejenom vytvářet kód, který řeší zadanou úlohu, ale měli bychom je učit i to, jak jej vytvářet co nejefektivněji a co nejefektivnější. Efektivitou vývoje přitom nemyslím jenom efektivitu vytváření kódu, ale především schopnost vytváření takového kódu, který nebude při pozdějších úpravách přítěží, ale bude navržen tak, aby byl co nejsnáze modifikovatelný a co nejjednodušeji udržovatelný.

5 POMOCNÉ NÁSTROJE

Prozatím jsme se však zabývali pouze vlastním programováním a jeho výukou. Součástí práce současného programátora je však i používání nejrůznějších knihoven, frameworků, kontejnerů a dalších externích programů. Podíl nejrůznějších použitých knihoven a frameworků se každým rokem zvyšuje a vlastní přínos programátora je v celém kontextu zdánlivě zanedbatelný – alespoň měřeno mechanicky počtem řádků kódu. Skutečný přínos pro správnou funkčnost celého kompletu je však stále významný a nenahraditelný.

Tak jak se zvyšuje podíl těchto knihoven a frameworků na řešení zadaných úloh, tak se přirozeně zvyšuje i podíl času programátora, který musí věnovat tomu, aby jeho program správně používal použité knihovny, aby se správně spolupracoval s požitým frameworkem, aby jej správně začlenil do použitého kontejneru atd. atd. Tuto práci často výrazně komplikuje nedostatečná nebo špatná dokumentace použitých nástrojů a v řadě případů i jejich nevysoká kvalita.

Podíl výše zmíněných aktivit na práci související s vývojem projektu se sice stále zvyšuje, ale ve výuce se tento rostoucí podíl zatím nijak neprojevuje. Tu tam se objeví snaha naučit studenty používat jednu konkrétní knihovnu, framework, kontejner, aplikační server, pomoc-

ný program apod. Neustále se však objevují knihovny, frameworky, kontejnery, servery a pomocné programy nové a často o generaci lepší než jejich předchůdci.

Mnozí cítí potřebu otevření předmětu, který by studentům poskytl nejenom všeobecný přehled o současných nástrojích toho kterého druhu, ale ukázal jim na těchto nástrojích také to, čeho si mají při výběru nového nástroje všímat a jak mají při prosazování vybraného nástroje argumentovat. Předmět, který by jim mimo jiné ukázal, že jednoduchost vývoje demonstračního projektu typu *Hello World* ještě zdaleka nemusí znamenat i jednoduchost vývoje skutečného projektu pro skutečného zákazníka se všemi jeho speciálními požadavky.

Je zřejmé, že příprava náplně takovéhoto předmětu nebude jednoduchá, protože v něm bude třeba seznámit studenty nejenom se současnou situací, ale i se současnými trendy a připravit je na to, aby dokázali vhodně zareagovat i na novinky, které se objeví až po jejich nástupu do praxe.

6 ZÁVĚR

Článek shrnul některé historické předěly v programování a současně upozornil na skluz, který má zařazení výuky nových postupů a trendů do výuky. Připomněl současný stav a upozornil na zdroje, v nichž je možno získat informace o nejnovější metodice výuky označované *Design Patterns First*. V závěru se pak zamyslel nad novými trendy v programování a nad dosavadní ignoraci těchto trendů v učebních plánech škol. Upozornil pak na potřebu zavedení předmětu, který by studenty na tyto nové trendy připravoval.

LITERATURA

[1] DRBAL Pavel, JÍLKOVÁ Helena: *Objektové a strukturované programování*. Celostátní konference Tvorba Softwaru '98 – příspěvek lze stáhnout na adrese http://formular-ekf.vsb.cz/formulare/F01/tsw/?page=prispevek&rok=1998&prispevekid=632

[2] JÍLKOVÁ Helena: *Objektový versus strukturovaný přístup při analýze a návrhu informačních systémů*. Seminář *Programování '85* – příspěvek lze stáhnout na adrese http://formular-ekf.vsb.cz/formulare/F01/tsw/?page=prispevek&rok=1993&prispevekid=490

[3] PECINOVSKÝ Rudolf: *Early Introduction of Inheritance Considered Harmful*. Objekty 2009, Hradec Králové.

[4] PECINOVSKÝ Rudolf: *Metodika Design Patterns First – Jak lze drobnou změnou výkladu zlepšit pochopení některých objektových konstrukcí*. Sborník konference DidInfo 2010. ISBN 978-80-8083-952-9.

[5] PECINOVSKÝ Rudolf: *OOP – Naučte se myslet a programovat objektově.* Computer Press 2010. ISBN 978-80-251-2126-9.

[6] PECINOVSKÝ Rudolf: *Skvělý program nemusí být skvěle prodejný*. Sborník konference Objekty 2007.

[7] PECINOVSKÝ Rudolf: *Výuka OOP žáků základních a středních škol*. Sborník konference Objekty 2003, Ostrava, ISBN 80-248-0274-0.

[8] PECINOVSKÝ Rudolf: *Proč a jak učit OOP žáky základních a středních škol*. Žilinská didaktická konferencia, 2004, Žilina.

[9] PECINOVSKÝ Rudolf: *Metodika výuky programování na rozcestí*. Sborník konference Poškole 2007, Lázně Sedmihorky.

[10] PECINOVSKÝ Rudolf: *Výuka programování pro praxi*. Sborník konference Informatika XXI 2008, Lázně Luhačovice.

[11] PECINOVSKÝ Rudolf: *Mýty ve výuce programování a metodika Design Patterns First* – zvaná přednáška. Sborník konference Objekty 2008, Žilina.

[12] TVRDÍK Josef: *Modulárně nebo strukturovaně?* Seminář ***Programování '85*** – ke stažení na http://formular-ekf.vsb.cz/formulare/F01/tsw/?page=prispevek&rok=1985&prispevekid=262.