30 LET STRUKTUROVANÉHO PROGRAMOVÁNÍ

J. Hořejš

1. REMINISCENCE A KONFESE

Nebývá zvykem psát odborné články v první osobě jednotného čísla. Poněvadž jsem však byl požádán o osobní pohled a nechci se vyhýbat jisté zodpovědnosti za popularizaci idejí SP, dovolím si udělat výjimku.

[Teorie a praxe] Hned z počátku se přiznám: ani jednoduché školní programy, které jsem napsal, ani těch pár složitějších projektů při nichž jsem asistoval, mě zdaleka neopravňují zařadit se mezi zkušené programátory. Mou sférou zájmu je teorie. Na svou obhajobu zde uvádím, že společně s praktiky nemám rád takovou teorii, která dělá z nouze ctnost a umělými konstrukcemi nahrazuje svou neschopnost vidět a řešit dobře motivované problémy. Několikrát se mi však stalo, že obecnost teorie mi pomohla udílet (dokonce užitečné) rady podstatně zkušenějším praktikům - a zážitky tohoto druhu sbírám náruživěji než filatelista vzácné známky. Dobrou teorii zato rád hájím mezi praktiky, kteří si často ani neuvědomují jak velkou mají pravdu když prohlašují, že s teorií mají nejvýš problémy jim nepochopitelné. Například bez onoho často posmívaného Turingova stroje by jeho uzemněná von Neumannova parafráze vznikla rozhodně později než se stalo. Praktik jeví tendenci pohrdat relikty teoretických koncepcí bez ohledu na jejich původní význam (částečně je to ovšem důsledek toho, že teoretik se je naopak snaží zbožštit), zatímco nad původním Bellovým telefonem či Edisonovou žárovkou stojí v němém obdivu. Pokusím se naznačit, že tento jev je možno pozorovat i ve sféře SP.

[První řídící struktury] Přesně před 30 lety, v r. 1954, vyšel článek - kniha A.A. Markova: Teorija algoritmov (Trudy matem. instituta Stěklova, XLII). Z hlediska computer science je tato práce pozoruhodná hned v několika směrech. V době, kdy počítače převážně ještě "jen" (numericky) počítaly, je zde podána (prakticky upotřebitelná, jak ukázal další vývoj!) teorie jistého druhu textových editorů, a to v jistém smyslu nejobecnější možná. Způsob "programování" Markovových algoritmů je zcela netypický z hlediska současného programátorského myšlení (mj. jde o dobrý test, jak snadno je kdo schopen přejít k nezvyklé formulaci algoritmu: Kdo z vás, prvotřídních řidičů, by byl schopen efektivně a bezpečně řídit auto, u něhož by byla vyměněna pozice pedálu spojky, brzdy a akcelerátoru?!). Na

druhé straně je explicitně dokázána uzavřenost modelu vzhledem k složení, větvení a iteraci, tj.: jsou-li A,B,C tři "normální" (= v jazyku této teorie vyjádřitelné) algoritmy a P libovolné slovo z jisté abecedy, jsou normální i algoritmy S : S(P) = B(A(P)); V : V(P) = $\begin{cases} A(P) & \text{je-li } C(P) = \Lambda \\ B(P) & C(P) \neq \Lambda \end{cases}$ (Λ je prázdné či jiné pevně dané slovo); I : $I(P) = A^n(P) = A(...(A(P))...)$, kde $C(A^n(P)) = \Lambda$ ale $C(A^{n+1}(P)) \neq \Lambda$. Markov nebyl programátor, z teoretických úvah však přirozeně (a to v jinak dost "nepřirozeném" modelu) odvodil tři základní idee struktury. (Vzdáme-li se výslovného zaměření na algoritmizaci, vystupujeme prvky zmíněných struktur dokonce v definici rekurzivních funkcí, ještě o 20 let starších - v pracích Gödela, Kleeneho a dalších matematických logiků.)

2. CO JE "STRUKTUROVANÉ"

Jednou z "námitek proti SP" je, že pod speciálním terminem prodává ledacos, podle toho co se tam hodí. K tomu:

Terminologie: Většina oborů se smířila s faktem, že původně zavedené termíny začínají žít vlastním životem bez ohledu na původní etymologii. Řeknu-li že mě zlobí auto, máiokdo v tom bude spatřovat sebekritiku, ačkoliv původní řecký význam vede spíš k této interpretaci (a Sókrates by to tak určitě vzal). Dokonce mám pocit, že většina z vás tvrdí, že pracuje s počítači spíš než se stroji na zpracování informací (jakkoliv malý podíl numerických výpočtů může veše zařízení provádět). Tedy i u oborů, které se rozvíjejí tak rychle, že původní jazykový význam termínů je stále ještě v dohledu, jsme schopni tolerovat jiný nebo aspoň značně zúžený lingvistický význam řady slov. Proč tedy trvat na tom, že SP musí proklamovat pouze a jen zásady "dobré strukturace", navíc jedině řídících struktur - a to jedině proto, že v době, kdy se název vynořil, byly otázky řídících struktur pro podstatně obecnější metodologii SP v ohnisku zájmu jako prakticky a teoreticky nejdůležitější?! Nezapomeňme, že strukturace dat, algoritmizace jako takové, pracovních týmů atp. - to jsou problémové okruhy, které vznikly později. Spíš je možno se divit, že mnoho dalších později zaregistrovaných složek obecné metodiky SP má opravdu co činit se strukturou! Ale dohodněme se raději, že termín SP vezmeme prostě jako terminus technicus.

Co to tedy je SP? Nejpohodlnější ovšem je definice pragmatická: SP zahrnuje všechny metodické postupy a odpovídající jazykové prostředky, které ulehčují proces konstrukce spolehlivých, přehledných a tedy modifikovatelných programů. Je možno ovšem pokusit se např. o

Pro metodiku SP je	
PODSTATNÉ	nepodstatné
DŮLEŽITÉ	méně významné
NUTNÉ	do jisté míry nahodilé
.	.
.	.
.	.
podřídit atributy uvedené vpravo těm uvedeným vlevo:	
FUNKCE	struktura
ABSTRAKCE	konkretizace
IDEA	realizace
DISCIPLINA	"umění"
SYSTEMATIKA	metody ad hoc
OBSAH	forma
GLOBALITA	lokalita
"SHORA-DOLŮ"	"zdola-nahoru"
CÍL	prostředky
.	.
.	.
.	.

taxativní výčet různých zásad, aspektů či hypotéz, sledující tento definiční cíl. Uvedené porovnání (tabulka) se o to pokouší na úrovni filosofických kategorií a představuje tak cvičení z dialektiky. Vzhledem k termínu "struktura" dospíváme dokonce k jistému "paradoxu": struktura je méně významná než funkce, kterou zachycuje. To je ale v pořádku! Strukturované programování klade důraz na volbu dobrých struktur: takových, které odpovídají funkční dekompozici problému, přehlednosti produktu, jež vyplývá z jeho systematického návrhu shora dolů etc. "Strukturovaný" je návrh tehdy, podřídí-li se omezením na vhodné struktury. Je usměrňován tím co se nemá odehrávat na úrovni atributů z pravé strany tabulky ve prospěch strany levé. V tomto smyslu jsou vlastnosti uváděné vpravo důležitější a charakteristické pro SP. (Podobně jako právní nařízení, i zde se mluví spíš o tom co se nesmí; tabulky "Kouření dovoleno", "Vstup třeba v teplácích" se obvykle nevyvěšují). Důležitý přínos a charakteristika SP spočívá již i v samotném zaregistrování a vymezení (relativních) hranic mezi funkcí a strukturou a dalších vztahů z tabulky; "klasické" SP je tak přirozeným odrazovým můstkem pro další vývoj - například princip abstrakce dospěl mezitím k dalšímu kvalitativnímu zlomu v pojmu abstraktních datových typů (ADT) apod.

V principu bychom mohli zkusit ve smyslu této diskuze navrhnout pro SP jiný termín, "přímý": funkční, systematické, ideové, cílově zaměřené... . Potíž je v tom, že (a) jediným slovem je tak jako tak stěží možno zachytit nejvýraznější charakteristiku - jde o celý komplex zásad; (b) prosazení (prosazování) nové terminologie je vždy obtížné, často pak vede k zmatku většímu než byl na počátku. Proto osobně nedoporučuji o žádné terminologické změně uvažovat; spíš by mi nevadilo zahrnout pod "SP" i řadu metod, které přesahují původní jeho vymezení jak bylo zmíněno při odvolávce na ADT a bude rozebráno dál.

3. VÝVOJ SP

Vývoj idejí, které daly vznik SP, přerostly jeho počáteční výměr a postupuje dál, zhruba podle linie

JAK → CO → JAKÉ CO → KTERÉ CO

[JAK] Programátoři prvních generací řešili poměrně jednoduše a jasně specifikované úlohy (numerické povahy) a "CO" tak zcela přirozeně ustupovalo problému "JAK". Programovatelné kapesní kalkulačky a jednoúčelové mikroprocesory nás ostatně i teď často vracejí k témuž problému: jsme-li krutě vázáni malou pamětí, specifickým operačním kódem a máme-li naopak relatívně jednoduše formulovaný problém (i když může velmi často být jednoduchý jen zdánlivě!), i ten nejhorlivější zastánce SP je nucen opustit (aspoň některé) jeho zásady a začít vymýšlet triky jak to tam všechno vepsat. Věřím ale, že úplné opakování chyb starých zlatých časů je i zde zbytečné a že jsou si toho vědomi i výrobci těchto nádherných hraček: "Křížové" kompilátory, vývojové stavebnice, vyšší programovací jazyky a jiné prostředky by se tak rychle nezmocnily této oblasti, kdyby to nebylo zaplaceno problémy, jež daly vzniknout SP.

[CO] Osobně věřím i tomu, že při "programování ve větším" (tedy pro klasické programování v "malém" i "velkém", jak se teď s oblibou říká) se většina z vás dožije doby, kdy CO bude jednoznačně důležitější než JAK. Že stojíme na dalším přelomu, kdy SP přeroste v "disciplínu programování", v níž se spojí klasické programování s metodami matematické (dynamické) logiky, proces tvorby programů s jejich současným ověřováním (přičemž tato Dijkstrova koncepce "verifikace" se bude bezpochyby v širokém okruhu programátorů prosazovat ještě setrvačněji než dnes již klasické principy SP), návrh programů s jejich přesnou specifikací pomocí jazyků, u nichž spoň forma bude zprvu přijatelná [mám namysli prostředky ADT, axiomatické specifikace, které představuje třeba Burstallův CLEAR nebo Bjørnerův (a jiných) META-IV - když už Jacksona a jeho podobné jsme již dnes občas ochotni považovat za odnož SP]. Věřím i v rozvoj prakticky upotřebitelných (polo) automatických systémů pro návrh, vývoj a transformace programů (třeba ve stylu mnichovského CIP).

[JAKÉ CO] Prosazování těchto idejí do praxe nebude zřejmě o nic lehčí - spíš těžší - než před deseti roky SP. Začít myslet v termínech specifikačních jazyků, které by měly nejprve donutit programátora, je-li zde ještě termín "programátor" vůbec na místě, aby jasně přesněji po dohodě se zadavatelem uvážil JAKÉ CO má vlastně vzniknout, jaké jsou komplexní požadavky na řešení problému, to vše bude zřejmě

vyžadovat mnoho teoretické, metodické i praktické práce.

[KTERÉ CO] To, co zatím jen tušíme - totiž z existence systémů, které přímo přesvědčí (třeba formou dialogu) nejkompetentnější instanci tohoto tématu, totiž samotného finálního uživatele, že vlastně nový co dělá a že ze všech přípustných možností to musí být on, který stanoví KTERÉ CO by bylo pro něj nejvýhodnější - to zatím přenechme oblasti sci-fi nebo aspoň rozvíjející se vědě kybernetických věd - umělé inteligenci, kterou zatím využíváme asi tak jak by amatér zahrádkář využíval solidně vyrobený počítač, kdyby měl náhodou tvar lopaty.

4. DOSAVADNÍ ZKUŠENOSTI SE SP

V tomto odstavci se zmíním o svém pohledu na SP v blízkém okolí. Jde o osobní názor, získaný někdy na základě dlouhodobějšího sledování, jindy jen prostřednictvím letmých rozhovorů či zběžné prohlídky výsledného programového produktu. Obecně je tedy "s ručením omezeným"; omlouvám se předem všem spolupracovníkům za formulace, které by oni sami shledali nepřiměřenými.

[Mimo pracoviště autora] Vázán slibem a osnovou, mohu se zde jen krátce zmínit o těch několika radostnějších chvílích, kdy jsem byl svědkem (resp. zván za svědka) řady pokusů většinou mladých odvážných jedinců z různých výpočetních středisek, kteří - v podstatě sami od sebe či jen z mírného doslechu - znovu "objevili" některé ze zásad SP [podobně jsem viděl na začátku počítačové éry u nás několik takových "objevů" zásobníku] jednoduše proto, že zaběhnaný postup a dělba práce v jejich středisku jim neumožňovala předkládat požadované produkty v požadovaných termínech. Radost smíšená s pocitem viny - tak bych asi charakterizoval své osobní pocity. Radost nad tím, že většinou (obvykle to začínalo problémy dokumentace, ale dostalo se i na další, včetně oněch klasických "řídících struktur" ve smyslu pokusů je legalizovat metodicky či vhodným technickým prostředkem - výměnou programovacího jazyka, jeho doplněním ve smyslu commentů, preprocesorů ap. je natolik užitečná, že si nakonec najde sama svou cestu. Pocit viny nad tím, že mnohé z prezentací SP byly zřejmě natolik zatíženy obecností a nedostatkem smyslu pro provozní problémy (vč. problémů terminologických), že toho "široké veřejnosti" zbývalo k doobjevování tak moc.

[Na pracovišti autora] ÚVT UJEP obnáší kolem 40 lidí, z nichž asi polovina by mohla/měla prosazovat/uplatňovat ideje SP v praxi. Jde o relativně malý kolektiv bohatě "strukturovaný": je zde skupina výzkumu (úzce napojená na katedru aplikované matematiky přírodovědecké fakulty, s níž spolupracuje i na výuce), skupina systémových pro-

gramátorů (k počítači EC 1033 a PDP(1/34), skupina ASŘ (řešící pro školu a resort školství problémy běžných agend propojených do systému AISŘ, skupina "vývoje" (spolupracující s praxí - například na softwarovém vybavení disketových pracovišť vyráběných Zbrojovkou Brno; další její členové propojují EC s PDP přes 8080, sestavili "univerzální" systém pro potřeby skupiny ASŘ aj.) a skupina aplikačních programátorů, podílející se na řešení některých výzkumných úloh jiných pracovišť školy (lékařské fakulty, katedry sociologie ap.). Většina členů tohoto kolektivu a většina učitelů "počítačové skupiny" katedry si - aspoň nepravidelně - najde čas na interní semináře výzkumného úseku, který nejenže akceptuje zásady SP, ale snaží se aktivně přispívat i k jejich pochopení a rozvoji, i k výuce navazujících tendencí, zmíněných v odst. 3. Zdálo by se tedy, že celá činnost ÚVT je prosáklá idejemi SP. Není tomu tak a dokonce není ani možné jednoznačně říct "tohužel". Totiž:

[Výuka] Formálně je vše v pořádku. Podle zásady "co se v mládí naučíš, ve stáří jako když najdeš" jsou proklamovány starší i novější principy SP od samého začátku studia, obsahově pak od důrazu na systematiku a uvědomělost při tvorbě softwaru až obecněji při jakékoliv činnosti až po ono ifthenelse a whiledo. Potřeba udělat si pořádek v těchto základních otázkách je jen podtržena očekáváním dalších změn v metodologii; změn, proti nimž klasické SP je dnes již stařenkou, kterou by nemělo být těžké dohonit. Prakticky je však třeba uznat, že někteří studenti jsou dost dlouho schopni nepochopit "ducha" SP a při nejbližší příležitosti jsou ochotni vysmeknout se jako řadu právě tak, jako třeba řádu kolejnímu. Stačí zápočtový či jiný časový stres a zápisky na programech rostou jak houby po dešti a systematika uniká každou skulinkou od etapy návrhu až po závěrečné testování. Nicméně lze doufat, že silnější jedinci překonají toto inkubační období a do praxe vstoupí aspoň částečně imunní proti nejhorším programátorským neřestem.

[Výzkum, systémové a některé aplikační programování] Zde je situace nejuspokojivější. Oba kolektivy mají dostatečné zkušenosti, teoretické i praktické, k tomu, aby byly schopny zcela přirozeně využívat tzv. SP od etapy specifikace až po implementaci, a to bez ohledu na to, zda pracují v prostředí, které SP favorizuje (např. se systémem RPS, či naopak při práci s asemblerem pod OS). Podle povahy problému při tom vystupují jako klíčové různá doporučení SP; jejich výběr resp. zdůraznění mohou při tom být dost subjektivní. Jednou bude prvořadé členění problému podle "principu abstrakce" s navazující metodou tvorby shora dolů a snahou o systematickou dokumentaci a dokumentované testování, jindy třeby přestavba algoritmu eliminující zbytečná

goto. Vcelku lze konstatovat, že na této úrovni se s metodou SP nesmažuje, že již vytvořily v krvi programátorů jisté ochranné látky, bez nichž by organismus asi brzy selhal.

(ASŘ a jiné aplikační programování) Na tomto poli se často projevuje známý fenomén - vazba na vysoce "nestrukturovaný" vnější svět vč. zmatků v projektech, specifikacích, časových limitech apod. To má za následek, že často vznikají produkty dost syrové, které srdce ortodoxního zastánce SP příliš nepotěší. Na druhé straně však nelze autorům upřít obeznámenost s principy SP a jejich produktům fakt, že je to aspoň někdy patrné. Osobně doufám, že v tomto mladém kolektivu potřebné metodické přístupy zrají a že přijde čas, kdy začnou účinněji pomáhat.

Závěrem odstavce opakuji preambuli celého tohoto odstavce: má "hodnocení" situace je osobní, téměř soukromé, určitě subjektivní: nebyly nasazeny žádné prostředky, které by aplikaci SP a její výsledky objektivně měřily.

5. ZÁVĚR

Shrnu-li výsledky svých pozorování i domnének, odvážím se konstatovat, že

- ideje klasického SP jsou natolik účinné a přirozené, že tam časem dospěje každý dobrý programátor (třeba pod jiným názvem a při různém akcentu jednotlivých zásad);
- ponévadž systematické propagace SP umožňuje tento proces urychlit a protože vnější svět bude ještě dlouho působit spíš jako retardační faktor, je třeba v šíření idejí SP pokračovat i když vstupujeme do období, kdy většina absolventů různých škol, vychovávajících programátory, by měla být s klasickou formou SP seznámena
- SP není fixní ideologie, ale základ pro vývoj progresivních programovacích metodik vůbec; necháme-li ho z moci definice zahrnovat i tyto nadstavby, je nutné na SP pohlížet jako na otevřenou soustavu, k níž je třeba i do budoucnosti přistupovat tak jak sama požaduje - systematicky.