Soukupův Grafický Preprocesor – první vykročení k systémům CASE – aneb zkusme programovat česky!

Bohumír Soukup

Motto: *„Ještě můžeš, ..., ještě můžeš, ... Tak, a teď se pojď podívat, co jsi udělal."*

1. Úvod

Snahy po zpřehlednění a zdokumentování programátorské práce se datují od samého počátku programování.

Vzhledem k tomu, že jsem za své praxe již poznal zhruba tisícovku programátorů s nimiž jsem více či méně podrobně rozebíral téma metodiky a způsobu programování (soustavně od roku 1976), domnívám se, že by mé poznatky a úvahy mohly být užitečné i dalším programátorům, analytikům, vedoucím týmů, učitelům programování a případným dalším adeptům programování v klasických jazycích.

K napsání tohoto příspěvku mě vedla situace ve výuce programování a v programování obecně.

Když občas poslouchám nebo čtu zasvěcené polemiky různých programátorů a metodiků o tom, který textový editor a který programovací jazyk je pro výuku programování ten nejlepší, nemohu se stále ubránit pocitu nesmyslnosti těchto diskusí.

Pro pochopení tohoto mého pocitu si představte spisovatele, který již několik let vlastní počítač, osvojil si již nějaký textový editor, či dokonce některý z programů DTP, a s těmito znalostmi a vědomostmi by měl poslouchat debaty svých kolegů–spisovatelů o tom, která značka psacího stroje a který jazyk je nejlepší pro napsání literárního díla.

Takže zde máme první dvě témata, kterým se budu věnovat ve svém dnešním příspěvku – nástroj a jazyk.

Třetím tématem je klasické „rozbíjení strojů". Tak jako se spisovatelé a konstruktéři zpočátku bránili nasazování počítačů do své praxe, brání se nyní i programátoři zavádění počítačů do jejich praxe. Ano jedná se o zavádění systémů CASE.

2. Textový editor – nástroj pro programátory?

Vzhledem k tomu, že program nevzniká bušením do klávesnice, ale analytickým myšlením, jak je všeobecně známo, nedá mi v posledních letech klid otázka, proč vlastně programátoři stále používají pro zápis algoritmu pouze textový editor a ještě na něm tolik lpí?

Proč například elektrotechnici, strojaři, stavaři, matematici, chemici a pod. se rádi naučí desítky různých značek a symbolů, aby je potom pracně ručně (dnes již pomocí počítačů) kreslili na papír. Nikoho z nich přitom ani ve snu nenapadne, aby svoje konstrukce, schémata či vzorce zapisoval pomocí textového editoru a ještě k tomu v nějakém cizím, umělém jazyce.

Na druhé straně přitom programátorům stačí k popisu jejich konstrukce–algoritmu pouze několik značek a z devadesáti procent nikoho z nich nenapadne, že by je měl používat, že by měl použít něco jiného nežli textový editor a umělý jazyk.

Že by snad programátoři nevytvářeli tak složité konstrukce, nebo snad jsou o tolik chytřejší a dokonalejší nežli jejich kolegové konstruktéři? Proč pak tedy tak dlouho svoje programy opravují, ladí a testují, a to při každé, i té nejnepatrnější změně?

Ne, řekl jsem si, a uklidnil se – neboť i já jsem občas také programátorem, a myšlenka, že bych tedy právě proto měl natolik převyšovat svoje kamarády konstruktéry mě ponékud zneklidnila. Vysvětlení bude určitě někde jinde.

TEXTOVÝ EDITOR!

Tento fenomén dnešního světa programátorů se pokusím ponékud blíže rozebrat, protože já osobně považuji při zavádění jednoduších systémů CASE pochopení problematiky textového editoru za klíčovou. Textový editor je to, co většinou každý programátor zná a ovládá ze všeho nejdříve a také nejlépe. Bohužel, mnoho „programátorů" ovládá pouze textový editor, v lepším případě i pár příkazů nějakého programovacího jazyka.

Textový editor má totiž zvláštní kouzlo. Jeho zvládnutím se z naprostého „neprogramátora" rázem stává PROGRAMÁTOR. Že o programování ještě vůbec nic neví? Nevadí – jeho „programy" mu již chodí (např.: PRINT „Ahoj světe"), takže je z něj SKUTEČNÝ PROGRAMÁTOR, a tak se i on může konečně zapojit do výše zmíněné diskuse, který z těch textových editorů a programovacích jazyků je ten pravý ořechový.

Když pak začne vytvářet ponékud komplikovanější programy, zjistí, že programování zase tak jednoduché není, ale naštěstí je zde všelék – DEBUGGER. Náš adept zajásá a po zvládnutí debuggeru, se stává ze SKUTEČNÉHO PROGRAMÁTORA PROGRAMÁTOR PROFESIONÁL.

A tak se textový editor a debugger stanou věrnými, ale často také bohužel jedinými pomocníky našeho „profesionálního programátora".

Že takovéto „neprogramátory" maskované znalostí nejnovějších textových editorů, kompilátorů a debuggerů, je i dnes někdo ochoten zaměstnávat a ještě jim platit, je sice podivné, ale je tomu tak.

Bedlivý čtenář již jistě pochopil, kam mířím. Ano, zkusme odbadnout, co se stane, když takovýmto „neprogramátorům" vezmeme jejich textový editor a nahradíme jej skutečným nástrojem pro programátory. Místo aby zajásali, že budou mít více času na analýzu, začnou kolem sebe prskat a škrábat, což je to za bibost, kdo si to zase vymyslel takový nesmysl, psát programy jinak nežli textovým editorem. Nic nepomůže vysvětlování a argumentace, že textový editor je nástroj pro písařky, že zapisovat algoritmy textovým editorem a ještě k tomu pomocí cizích slov nějakého umělého jazyka, je nejen značně neproduktivní, ale také činnost, nad jejíž pošetilostí se pousměje leckterý kolega – konstruktér.

Poznámka: Ono je opravdu dosti tragikomické, pozorovat vzdělaného odborníka, kterak umně ručně zapisuje a upravuje text programu tak, aby každé begin, if, repeat mělo odpovídající end, endif, until, aby všechny vnořené struktury začínaly a končily ve stejném sloupečku, aby každá struktura byla opatřena příslušným komentářem, pokud možno stejným na začátku i na konci struktury. Viděl jsem dokonce letité zkušené programátory, kteří ve chvílích nejvyššího zoufalství („očekáváno end, endif" apod.) vytiskli zdrojový text na tiskárně a pomocí pravítka se tento text snažili upravit alespoň tak, aby jim ho kompilátor mohl vůbec přeložit.

Martýrium, které nastává po prvním úspěšném překladu, a které jinak celkem normální lidé dobrovolně podstupují proto, aby program začal vůbec fungovat, je již asi zbytečné popisovat.

Jednu jízlivou poznámku si však přece jenom nemohu odpustit. Dnes se píše rok 1992!

O co tedy vlastně jde? Jde o to, že člověk „neprogramátor" není schopen bez textového editoru, programovacího jazyka a debuggeru vytvořit zhola nic. A protože si je této své „programátorské nahoty", někdy pouze intuitivně, vědom, brání se tak zarputile všem nástrojům, které jsou zaměřeny na oblast analýzy. A naopak, bouřlivě vítá každý sebepodřadnější nástroj, byť by uměl jenom naformátovat zdrojový text, případně zvýraznit begin a end.

2. Stručný popis SGP

SGP (Soukup Graphic Preprocessor) je jednoduchý nástroj patřící do skupiny lower CASE systémů. Je založen na technologii strukturovaného programování podle M.A.Jacksona.

Pomocí SGP se zdrojový text programu vytváří v následujících krocích:

1. Popis datových struktur a návrh programové struktury (funkční schéma programu, procedury).

2. Popis jednotlivých výkonných akcí a podmínek (seznam operací a podmínek).

3. Umístění označení výkonných akcí a podmínek do programové struktury.

4. Kódování výkonných akcí do zvoleného programovacího jazyka a doplnění deklarací pomocných proměnných.

Nyní je vytvořena programátorská dokumentace (soubor SGP).

5. Spuštění preprocesoru – výsledkem je zdrojový kód programu ve tvaru pro kompilátor.

Do takto vytvořeného zdrojového kódu programu se již neprovádějí žádné změny! Veškeré změny se provádějí zásadně pouze do programátorské dokumentace (souboru SGP). Zdrojový text programu je chápán pouze jako dočasný soubor, který slouží pouze jako vstupní soubor pro kompilátor.

V současné době se jedná zřejmě o nejjednodušší, nejuniverzálnější a také nejlevnější lower CASE systém na softwarovém trhu.

Integrované prostředí SGP obsahuje editor struktogramů, textový editor a jazykově orientované preprocesory. Od prosince 1991 je SGP k dispozici pro Pascal, C, C++, dBASE, FoxBASE, FoxPRO, Clipper, Modulu–2, Paradox, Fortran, Cobol a Informix.

Seznam článků s podrobnějším popisem systému SGP je uveden v seznamu literatury na konci tohoto příspěvku.

3. Dosavadní zkušenosti s SGP

3.1 Příchod SGP na československý trh

Premiéra uvedení SGP na trh proběhla na podzim roku 1988.

Programátorská veřejnost však zaznamenala příchod SGP až na jaře 1990, kdy naše firma SGP Systems měla již dostatek finančních prostředků na vlastní propagaci.

Příchod SGP na československý trh vzbudil mezi programátorskou veřejností značný zájem. Jednak to byla rozsáhlá reklamní kampaň, na kterou do té doby naše odborná veřejnost nebyla zvyklá, ale především málokdo z tehdejších programátorů byl spokojen se svou produktivitou práce, takže by rád získal něco, co by mu tuto jeho nízkou produktivitu pomohlo zvýšit.

3.2 Polarizace názorů

Okamžitě s příchodem SGP se začaly polarizovat názory programátorské veřejnosti. Jako první zareagovali odborníci. První skupina, mající teoretické znalosti a praktické zkušenosti z oblasti strukturovaného programování řekla bez výjimky jednoznačně „ANO".

Hned vzápětí se vytvořila tříčlenná skupinka kolem amatérského časopisu Bajt (6/1990), která řekla své jednoznačné „NE".

Podle mého názoru se tehdy jednalo pouze o snahu šéfredaktora zmíněného časopisu zvýšit čtenářský zájem o začínající časopis senzačním odhalením, na které si v Československu troufne pouze „nezávislý" Bajt. Jedině tak si lze zřejmě vysvětlit, proč šéfredaktor Bajtu nedal SGP recenzovat odborníkům na strukturované programování, jak původně slíbil, ale odborníkům snad na všechno ostatní, především pak na textové editory, kteří nikdy v životě Jacksonovu technologii nejen nepoužili, ale kterým se celá tato technologie scvrkla na nepodstatné „Jacksonovy obdélníčky".

Ing. P.Adámek, Soukup Graphic Preprocessor: „V recenzovaném programu se však z interpretace struktogramu ztratila i ta troška grafiky, a z obvyklých obdélníčků s nápisy zbylo jen torzo – právě jen ty nápisy."

MUDr.J.Kofránek, Cirkus bez manéže: Soukupův grafický preprocesor:
„Sám systém má s grafickou reprezentací algoritmů společné pouze jméno. Vychází z dnes již klasického Jacksonova způsobu znázorňování struktury algoritmů, ale se svým vzorem má málo společného. Místo Jacksonových obdélníčků nám Soukupův grafický systém poskytuje jakýsi textový čárový kryptogram, plný symbolů."

Není potom divu, že s takovouto znalostí JSP (Jacksonovo strukturované programování) potom došli autoři ke svým „senzačním odhalením".

Ing. P.Adámek, Soukup Graphic Preprocessor: „... Jsou to vlastně prkotiny ve srovnání s tím, že celá koncepce programu je na hlavu postavená a ve výsledku nejen že nesplňuje téměř žádnou ze slibovaných výhod, ale mnohdy dělá pravý opak."

MUDr.J.Kofránek, Cirkus bez manéže: Soukupův grafický preprocesor: „Hlavní a nejzákladnější chybou je podle mého názoru asi to, že si tvůrci dost dobře neuvědomili, k čemu a jak by vlastně měl tento systém sloužit. Poněkud problematická je totiž sama filosofie celého systému"

No a když se čtenářům (v té době jediného počítačového měsíčníku na pultech) dostane ještě ujištění o odborných a morálních kvalitách recenzentů, je pro laiky výsledek zcela jasný.

Šéfredaktor L. Zajíček: „Samozřejmě nedáváme programy recenzovat komukoli, ale profesionálům, u kterých si můžeme být jisti i etickým přístupem k věci."

Poznámka: Výsledná bilance našich i zahraničních posudků SGP byla po těchto „recenzích" (leden 91) 6 : 2, dnes je 12 : 2. Ve prospěch SGP.

3.2 Současná situace

V předchozí kapitole jsem se rozepsal podrobněji o reakci Bajtu na SGP pouze proto, že tyto reakce jsou zcela typické u programátorů, kteří dosud psali své programy pouze textovým editorem a používali zápis odshora dolů.

Ke stejným závěrům jistě došly i slovky dalších programátorů, kteří se pokoušeli vytvořit zdrojový text s pomocí SGP stejným způsobem jako s textovým editorem, ať to byli zkušení programátoři nebo začátečníci.

Začátečníci však mají situaci se zvládáním SGP ulehčenou v tom, že na textovém editoru ještě nejsou tolik závislí. Skutečný programátor (nezatížený) zvládne SGP za necelý den, zatíženému to trvá někdy i dva roky, neprogramátor nenapíše pomocí SGP ani řádku – nikdy.

SGP se nyní konečně dostává do podvědomí programátorské veřejnosti již ne jako zázračný lék, který z neprogramátora udělá rázem programátora, ale jako účinný nástroj pro tvůrčí programátory, vybavené alespoň minimálními teoretickými a praktickými znalostmi strukturovaného programování.

Zájem o SGP vzrůstá současně se vzrůstajícím uvědomování si důležitosti programátorské dokumentace.

V současné době máme v ČSFR zhruba 500 registrovaných uživatelů. Počet neregistrovaných lze těžko odhadnout.

3.3 Aktivity firmy SGP Systems

Vzhledem k tomu, že jsme zjistili, že největším problémem při zavádění SGP je neznalost metodiky strukturovaného programování, nabízíme všem zájemcům o tento systém nezávazné jednodenní seznamovací školení. Během tohoto školení se zájemci naučí ovládat SGP, dozví se základní principy strukturovaného programování a získají informace potřebné pro kvalifikované rozhodnutí zda SGP koupit či nekoupit.

Pro větší skupiny zájemců organizujeme předváděcí semináře.

3.4 Předpokládaný další vývoj

Abychom překlenuli počáteční potíže programátorů při přechodu na kvalitativně nový způsob práce, rozhodli jsme se, že k SGP vytvoříme debugger, který bude umět trasovat program přímo ve struktogramu. Tím by měla podle mého mínění padnout jakákoliv potřeba prohlížet a dekódovat zdrojový kód.

Debugger uvedeme do prodeje na podzimním veletrhu INVEX–COMPUTER '92.

Předváděcí verzi pro Turbo Pascal budeme mít k dispozici již na semináři Programování '92.

Co se týče rozšiřování SGP v Československu. Předpokládám, že především z řad soukromých firem bude stále větší hlad po kvalitních a přitom ukázněných programátorech, schopných pracovat v týmu na základě stanovené programovací technologie. Zde pak budou mít systémy jako je SGP jistě otevřeny dveře.

4. Shrnutí příčin, bránících širšímu nasazení nástrojů CASE

Na prvním místě stojí příliš lákavá snadnost programování. „Programovat" může vlastně každý kdo umí rozeznat jednotlivá písmena a číslice, v lepším případě číst (psát umět nemusí). Zatímco zkonstruovat sebejednodušší přístroj, vymyslet nový sebejednodušší vzorec vyžaduje alespoň minimální průpravu v daném oboru, stačí k napsání programu PRINT „Ahoj světe" pouze výše deklarovaná znalost čtení.

V souvislosti s tím stojí hned na druhém místě snaha výrobců kompilátorů po maximálním zisku. Ten docilují chytře tak, že nutí programátory psát své programy textovým editorem, protože jedině tak mají zaručeno (ti výrobci samozřejmě), že programátoři budou dělat neustále tytéž chyby z nepozornosti, které díky této „textové technologii" nebudou schopni sami snadno a rychle najít. No a takový programátor potom velice rád zaplatí za debugger, code view, flow chart a já nevím co ještě. Vím dobře, že toto moje konstatování mnohé programátory šokuje, mnohé pobouří. Pokud se jim však po uklidnění podaří nalézt jiné rozumné zdůvodnění proč v roce 1992 je dřívě převažujícím nástrojem pro zápis algoritmů právě nástroj pro písařky, velice rád se s tímto vysvětlením seznámím.

Na třetím místě pak stojí zdánlivá převaha programátorů – sportovců. Pro lepší vysvětlení tohoto pojmu používám často příměr z neprogramátorské oblasti.

Při nákupu jízdního kola můžeme pozorovat dvě skupiny lidí. Jedni, kteří si kupují jízdní kolo proto, aby se mohli dopravovat z místa „A" do místa „B" a druzí proto, aby denně našlapali alespoň 150 km. Těm druhým přitom nezáleží vůbec na tom kam jedou. Pro ně není kolo dopravní prostředek, ale sportovní náčiní.

Cyklisti – sportovci jakékoliv zmechanizování a zautomatizování šlapání nechápou a přirozeně odmítají. Pro ně není cyklistika prostředkem k překonávání vzdáleností ale prostředkem k posilování svalů.

A proč je převaha programátorů-sportovců pouze zdánlivá? Protože stejně tak, jako kolem sebe vidíme sportovně vystrojené a vyzbrojené nesportovce, jsou ve stejném poměru jsou kolem nás stejně vystrojeni a vyzbrojeni „neprogramátoři".

5. Volba programovacího jazyka

Podobně jako neučíme nejprve nikoho německy, italsky, francouzky kvůli tomu, abychom mu vysvětlili co to je román, povídka či báseň, nemá smysl učit kohokoliv nový (programovací) jazyk jenom proto, abych mu mohl vysvětlit základní principy algoritmi-

zace. Podle mého názoru a na základě mých vlastních zkušeností, když někdo neumí program „vymyslet" ve své mateřštině, nebude mu nic platná znalost Pascalu, „Céčka" ani 4GL.

Proto již také několik let poukazuji na nesmyslnost diskusí o tom, který programovací jazyk je nejlepší a který by se měl vyučovat na školách. Jsem přesvědčen, že pro výuku čehokoliv, informatiku a programování nevyjímaje, je nejlepší jednoznačně jazyk mateřský.

6. CASE ano či ne?

Pokud dnes někdo stále ještě váhá, zda by potřeboval CASE či nikoliv, nebo zda je již na jeho nasazení dostatečně připraven, mám pro něj jedno doporučení. Vyzkoušejte si nejprve SGP. Za setinu nákladů a času budete okamžitě vědět na čem jste a co vlastně potřebujete.

7. Závěr

Když si negramotný člověk koupí tužku, neznamená to ještě, že již umí číst, psát a počítat. A když se konečně naučí číst, psát a počítat, opět to ještě neznamená, že je z něj již automaticky spisovatel nebo matematik.

Teoretické i praktické zvládnutí systému CASE je pro jeho efektivní nasazení sice podmínkou nutnou, ale bohužel ne postačující. Při výběru vhodného CASE je proto potřeba mít na mysli, kdo s tímto systémem bude pracovat.

Literatura:

[1] Jackson, M.A.: Principles of program design. Academic Press, New York, 1975.

Články:

[2] Soukup, B.: Soukup Graphic Preprocessor. MAA 3/90.

[3] Vaníček, J.: Strukturované programování. Otázka zní: Je nutné zavádět do programování pevný řád?. E & My N/90.

[4] Zajkr, B.: Nejjednodušší systém pro podporu strukturovaného programování. Tech.týdeník 41/90.

[5] Soukup, B., Hajný, P.: Soukupův grafický preprocessor aneb jak se (málem) nestát milionářem – interview. CW 13/90.

[6] Ostatnický, K.: Soukup Graphic Preprocessor. CW 15/90.

[7] Adámek, P.: Soukup Graphic Preprocessor. Bajt 6/90.

[8] Kofránek, J.: Cirkus bez manéže: Soukupův grafický preprocesor. Bajt 6/90.

[9] Řepa, V.: SGP – Soukupův grafický preprocesor. Softwarové noviny 1/91.

[10] Dostál, J.: SGP? ... SGP!. Softwarové noviny 1/91.

[11] Soukup, B., Koubský, P., Babenský, Z.: Chce to myslet trochu jinak – interview. Softwarové noviny 1/91.

[12] Pecinovský, R.: Soukup Graphic Preprocessor: chválený a zatracovaný. P+C 3/91.

[13] Soukup, B.: Reakce na recenzi. P+C 3/91.

[14] Hron, M.: JSP: Jackson–Soukupovo programování. Bajt 4/91.

[15] Pecinovský, R.: Ještě jednou SGP. Bajt 4/91.

[16] Soukup, B.: Soukup Graphic Preprocessor. Eastern european computer law information, 1/91, Aura–Pont.

[17] Pecinovský, R.: SGP a JTP (Jacksonova technologie programování). CHIP 1/92.

[18] Pecinovský, R.: Stále vpřed. Recenze na SGP 2.70,

[19] Řepa, V.: Srovnatelné pouze ceny. Srovnávací recenze SGP a S–editoru. Softwarové noviny 2/92.

[20] Pecinovský, R.: Dva prostředky pro podporu programování. Srovnávací recenze SGP a S–editoru. Softwarové noviny 2/62.

Autor: Ing. Bohumír Soukup
SGP Systems
Masarykovo nám.21
686 01 Uherské Hradiště
tel/fax (0632) 3507