Jiří Boleslav, TOS Kuřim, Jan Chlouba p.m., VÚMS Praha, Petr Jiříček p.m., OKD AŘ k.ú.o., Ing. K. Kopčil, Vítkovické stavby Ostrava, Ing. Egon Kratochvíl, VÚMS Praha, Ing. Zdeněk Masařík, OKD AŘ k.ú.o., Ludvík Nováček, OVS VHJ Sigma Olomouc, Ing. Bohumil Ševčík, OKD AŘ k.ú.o., Ing. Vojtěch Tloušť, Čokoládovny Modřany, RNDr. Josef Zelenka, VSŽ Košice Pavel Šimoník

PROBLÉMY VYUŽÍVÁNÍ ROZHODOVACÍCH TABULEK – [illegible]

S problematikou rozhodovacích tabulek se již řadu let čas od času setkáváme v odborné literatuře. Články a knihy o rozhodovacích tabulkách se shodují ve výčtu výhod, které rozhodovací tabulky poskytují. Umožňují úplné, přehledné a úsporné formulace zadání programu, větší záruka logické správnosti věcného obsahu, postižení všech reálných kombinací uvažovaných podmínek, usnadnění vzájemné komunikace mezi programátorem, analytikem a dokonce i uživatelem, lepší a rychlejší orientace ve složitějších algoritmech, to jsou zhruba hlavní výhody uváděné k metodě rozhodovacích tabulek. Ale to jsou také vlastnosti, které snad nejvíce ze všeho dnešní programování a analýza potřebují. A tu vidíme, že rozhodovací tabulky jsou rozšířeny mnohem méně, než bychom čekali. Kromě několika výjimečných pracovišť se buď nepoužívají vůbec nebo jsou záležitostí jednoho či dvou nadšenců z celého podniku.

Na podzim roku 1977 se na výzvu pořadatelů tohoto semináře sešla skupina lidí, kteří projevili zájem a také měli co říci k problematice využívání rozhodovacích tabulek.
Byly předloženy tři otázky:
- jak to, že se rozhodovací tabulky přes své proklamované přednosti tak málo používají?
- je žádoucí, aby se masově používaly?

- jestliže ano, co je třeba pro to udělat?

Diskuse, která vznikla, byla částečně ústní a částečně písemná (vnější korespondence). Tento článek je zpracováním vzniklých materiálů.

Navržená témata se sdílejí se tří, na sebe sice navazujících, leč samostatných otázek. Proto se pokusím stručně vyjádřit svůj názor na každou otázku odděleně.

1. Proč není metoda RP přes své prokazované přednosti dosud u nás masově používána mezi programátory a analytiky?

Je bohužel skutečností, že se u nás dosud stále neprosazují RP v míře, která by odpovídala významu a přednostem této metody. Tato skutečnost se stane zvlášť výraznou, porovnáme-li rozsah nasazení RP u nás s rozsahem praktického využívání RP v jiných vyspělých státech. Tato situace je po mém soudu způsobena především:

a) Nehlubokými znalostmi principů a aplikačních možností RP, jakož i malými zkušenostmi s využíváním RP v projekční a programátorské práci. Domnívám se, že je to způsobeno mj. tím, že současná generace programátorů a analytiků je zvyklá na jiné metody a nechtějí z všemožných důvodů potřebu hledání nových metod. A nastávající programátoři a analytici??? Kde se vlastně mohou používání RP naučit?
Ve škole? V kurzech? Někde tam snad základům, ale pak přijdou do praxe a tam se setkají se svými staršími kolegy, kteří jim brzy "vmatí" svůj, tj. klasický styl práce.

b) Nevybaveností drtivé většiny počítačů našich uživatelů programy, které umožňují automatickou konverzi RP do cílových programů. Přes všechny nesporné výhody RP při analýze, dokumentování algoritmů a jejich ručním převodu do programů, spatřuji hlavní z hlediska budování ASŘ i nejperspektivnější vlastnost RP v tom, že při vhodné aplikaci mohou sloužit jako univerzální komunikační prostředek o rozhodovacích procesech libovolného druhu mezi uživatelem, analytikem, programátorem a počítačem. Troufám si dokonce

tvrdit, že RT mohou za určitých okolností umožnit uživateli komunikovat přímo s počítačem, aniž by ovládal programovací jazyk do všech podrobností a aniž by věděl jak se úloha v počítači řeší. Spojením RT s počítačem se tedy v jistém smyslu otevírá cesta k racionalizaci programovacích a projekčních prací. Existence programového vybavení, umožňující automatické zpracování RT na počítači by konečně, byť nepřímo nutila (alespoň tu převážnou část) programátory a analytiky k tomu, aby to s těmi tabulkami přeci jen zkusili.

c. Tím, že na rozdíl od porovnatelných projekčních a programovacích a výpočtových organizací v kapitalistických státech, se u nás (snad proto, že zde neexistuje ostrý konkurenční boj a honba za maximálními zisky) zdaleka nepožaduje striktní, rychlé a bezpodmínečné odstranění a využívání všeho, co slibuje zvýšit racionalizaci a tím i produktivitu projekčních a programovacích prací.

d. Odlišnou metodikou tvorby programů. Sekvenční (lineární) vypracování jednotlivých kroků při němž je testování podmínek prokládáno výkonnými příkazy - což je typické pro vývojové diagramy - je RT nahrazeno komplexním (plošným) nazíráním na řešený problém. Pořadí testování podmínek a realizací pravidel se tu nevěnuje téměř žádná pozornost. Vzhledem ke konzervativnosti programátorů (a je překvapující jak může být někdy hluboká), pokud jde o metodiku programování, se proto stává, že první reakce na RT nebývá právě příznivá, neboť aplikace RT znamená vždy zásah do dosavadních zvyklostí a návyků.

5. Je žádoucí, aby byla metoda RT masově používána?

V každém případě ANO! Úkoly spojené se zaváděním ASŘ, tj. systémů řízení, které by kvalitativně zvyšovaly úroveň řízení na hlavních hierarchických stupních řízení podniku jsou opravdu náročné. A každá metoda, která nám k tomu dopomůže, by se neměla opomíjet. To, že RT takovou metodou jsou, se zdá

být mimo jakoukoliv pochybnost. Vždyť vyjdeme-li z publikovaných zahraničních zkušeností, pak je až překvapující jakých výsledků bylo dosaženo. Tak např. anglická firma ICL uvádí, že výpočtová střediska, kde se přešlo na využívání RT v projekci a programování, zaznamenala růst produktivity práce o 30 - 80 %. Dobré zkušenosti s využíváním RT byly publikovány též v NDR.

3. Co je třeba pro to udělat?

a) Vybavit výpočtová střediska našich uživatelů účinnými překladači RT, schopnými převádět algoritmy rozhodovacích procesů do vyšších programovacích jazyků. Pro čsl. počítač 3. generace EC1021 již takový překladač existuje. Jmenuje se PROTAB a VÚMS, který je řešitelem popř. organizace ZOTO, jsou připraveni jej poskytnout každému, kdo o to projeví zájem. Rovněž v současné době vyvíjený počítač EC 1025 bude obdobným překladačem vybaven.

b. Propagovat, popularizovat a školit metodu RT. Problematiku využívání RT v projektování a programování by se daleko širší měly zabývat odborné školy, organizace zabývající se prodejem a instalací počítačů (KSNP, DATASYSTÉM), ČSVTS aj. První kroky již byly vytvořeny. Tak např. VŠE zařadila RT do svých osnov, KSNP organizují ve školním r. 1977-78 celou řadu kursů, kde se o metodě RT hovoří a konečně i existence této skupiny zájemců ukazuje, že i VTS chápe své místo v tomto snažení.
Tato činnost je však podle mého názoru nutné ještě prohloubit, rozšířit a to hlavně - dotáhnout až do praktické aplikace na podnicích.

Na základní problém: proč se metoda RT přes své (podle mého názoru nesporné) výhody dosud u nás masově nepoužívá, není jednoduchá odpověď. Domnívám se, že jde o celou řadu příčin, z nichž nejhlavnější jsou asi čtyři. První příčinou je neznalost metody a jejích výhod, případně pouze nedostatečné

neznalosti, které způsobí při použití metody potíže a nakonec odradí od jejího používání. Druhou příčinou je přirozená setrvačnost lidí, která vždy brzdí přijetí nových metod a způsobuje, že mají-li lidé sami uznat přednosti nové metody a na základě toho přejít od používání staré zaběhnuté metody, která do té doby dostačovala a vyhovovala, k používání nové metody, musí být nová metoda nejen jednoduše lepší, nýbrž musí být o mnoho lepší, a tento fakt musí být navíc zřejmý a prokazatelný.

Neznalost metody RP by bylo možno odstranit celkem snadno, neboť metoda není složitá a každý programátor ji v krátkém čase lehce zvládne. Avšak zkušenost ukázala, že odstranění první příčiny - neznalosti - nestačí. <u>Překonání konzervatismu je důležitější a obtížnější.</u> Druhou příčinu lze téměř zredukovat na soupeření mezi vývojovými diagramy a metodou RP. V tomto soupeření má metoda RP jedinou zbraň - své přednosti, a ty nejsou vždy zcela zřejmé. Naproti tomu vývojové diagramy jsou již zavedené a vyhovující pro dané aplikace. Navíc mají na své straně setrvačnost lidí, a ta nakonec rozhodne v neprospěch metody RP. Z toho plyne, že přesvědčovat hotové programátory a analytiky je obtížné.

Pro širší používání metody RP by asi bylo vhodné seznamovat s ní programátory a analytiky již během jejich vzdělávání a odborného růstu, paralelně se základy programování a metodou vývojových diagramů. Tak by vývojové diagramy ztratily svůj náskok a v soupeření obou metod by rozhodly výhody a nevýhody. Pravděpodobně by se však nakonec používaly obě metody, každá tam, kde je výhodnější. A to by byl podle mého názoru ideální stav a cíl, k němuž by mělo směřovat úsilí o prosazení metody RP do programátorské a analytické praxe.

---

1. Proč není metoda RP dosud u nás masově používána?

Já rovněž první příčinu spatřuji v neznalosti. Metodě RP se nevyučuje na vysokých školách, informace o teorii RP

s praktickými zkušenostmi a využíváním RT nemají dostatečnou publicitu.

Druhá příčina je v tom, že RT jsou značně srozumitelné (pasivní znalost lze získat během 30 minut), což vyvolává nekritické nadšení a podceňování složitosti.

Třetí příčina je v tom, že sestavení RT vyžaduje zcela jiný myšlenkový postup, než na který jsou analytici a programátoři zvyklí při sestavování vývojových diagramů. Snažit se formulovat podmínky a respektovat jejich časovou souvislost je náročné a vyžaduje trpělivé vypracovávání řady příkladů. Neúspěchy dovedou slabší jedince odradit.

Ti, kteří vytrvají a techniku RT si dobře osvojí, jsou překvapeni, když pomocí RT nalezli chyby ve složitých vývojových diagramech, které kdysi sami vypracovali. Jejich nadšení pak vede k tomu, že se snaží všechny problémy řešit pomocí RT.

A v tom spočívá čtvrtá příčina: RT jsou v zásadě vhodné pro všechny činnosti, závislé na určitých "pravidlech hry". Jejich použití je výhodné zpravidla jen tehdy, jestliže obsahují nejméně tři jednoduché podmínky, které časově souvisejí, příp. když jsou nutná více než tři pravidla. V běžných dílčích je těchto případů poměrně velmi málo, v převážné míře se vyskytují RT s jedinou nebo dvěma podmínkami. Proto pokusy o popis logiky celého programu pomocí RT spočívají zklamání a vedou někdy ke krajnímu návratu k vývojovým diagramům.

V literatuře se jako další příčina nedostatečného využívání RT uvádí nedostatek vhodných překladačů. S tímto názorem nesouhlasím, používání podle mých zkušeností lze RT účinně využívat i bez překladačů.

2. Je žádoucí, aby byla známky používána?

Rozhodně ano. Přednosti RT jsou nesporné:

a) ve vztazích k odborným útvarům slouží mj. jako prostředek k lepšímu dorozumívání, používání
- jednoznačně a přehledně přiřazují příslušné činnosti ke

všech možných kombinací podmínek,
- umožňují kontrolu, že byly vyčerpány všechny možné kombinace podmínek,
- umožňují zjistit a odstranit rozpory a redundance,
- umožňují modulární skladbu RT rozdělením problému na dílčí části a tím usnadňují postupné řešení problému i přehlednost RT;

b) odstraněním logických chyb ještě před programováním usnadňují programování a navíc usnadňují dohledání logických a věcných chyb podle jednotlivých pravidel;

c) standardizovaná forma je vhodná pro dokumentaci, je srozumitelná a čtivá.

3. Co je třeba pro to udělat?

Techniku RT, starou 20 let, by měly vnést do vědomí i naše vysoké školy, které mnohdy dovedou velmi rychle reagovat na technický pokrok.

Dobrá hodinová přednáška o RT je lepší než špatná nebo žádná. Nemůžeme ovšem od ní očekávat, že naučí více než pasivní znalosti RT; v každém případě by však posluchači měli být též upozorněni na úskalí, která je čekají.

Základní změnu ve využívání RT může přivodit jen vhodné déletrvající školení. S ohledem na různou úroveň počítačních znalostí považuji za nejvhodnější formu _dálkové_ školení, kdy posluchači dostanou příslušné materiály, obsahující kromě nezbytné teorie řadu vyřešených příkladů, postupně od nejjednodušších až po nejsložitější, jakož i způsoby programování (Pollackův algoritmus, Veinottův algoritmus a pod.). Dobrým základem je na př. publikace /15/. Na závěr by měl být zadán složitější příklad, který by byl projednán na závěrečném soustředění účastníků. Snad by bylo dobré, aby takový kurs byl zaměřen nejen na RT, nýbrž i na strukturované a normované programování, aby tyto moderní racionalizační metody vešly v širší známost a do běžné praxe.

Hlavní důvod, proč RT nejsou více rozšířeny, spočívá v tom, že jazyk RT je orientován na programátory. Programátor, který programuje na př. v Cobolu, často sám nezná celý arsenál toho, co tento jazyk umožňuje, většina si udělá jakousi svoji podmnožinu příkazů, se kterými prostě vystačí. Přesto, že toto má vyzkoušené, ví, co toto dělá. A více nebo méně to tomu programátorovi stačí. Pro takového člověka je velkým problémem, když on se nenaučil ještě celý Cobol, proč by se měl učit ještě nějaké RT, když neví to, co dokázal popsat těmi RT, dokáže popsat už tím aparátem, který zná. Takže se domnívám, že zásadní nedostatek je v tom, že jazyky RT jednak jsou určovány programátorům, domnívám se, že by to mělo být směřováno na analytiky. To znamená, že by to mělo být ne nějaké rozšíření stávajících jazyků, ale úplně samostatné jazyky, které slouží v podstatě analytikovi nebo respektive pomocí nichž analytik může formalizovanou formou napsat program. To znamená obejít programátora a vyřadit programátora jako dejme tomu nechci říci třídu. Jinak, řekl bych to obrazně, to připomíná situaci, kdy budu vyrábět třeba košíky a ty košíky budu nabízet medvědům. Programátoři mají svoje jazyky, a proto tedy ten jazyk RT těžko může u nich prorazit. Jsou výjimky, že některý programátor uvidí RT, najede u Fukse, to je něco pro mne, ale já myslím, že to jsou jenom výjimky.

---

1. Programátor trpí chorobou z povolání: myslí v tom jazyku, ve kterém programuje (Cobol, Assembler, PL/I, Fortran ..), když chce něco zalgoritmizovat. Tedy při algoritmizaci i těch nejjednodušších rozhodovacích situací myslí v kategoriích if-then-else nebo if-then. Tak se to naučil každý z nás v nějakém kursu nebo praxi. Učil se nějaký jazyk, tam se naučil algoritmizovat právě tímto způsobem a ne jinak. Dále nemá k dispozici přímo zápis RT do programu - pokud někde ta možnost je, tak se toho příliš nepoužívá, neboť tomu brání dejme tomu přílišný konzervatismus daného člověka. Proč by se to učil a ztrácel čas zvládnutím

nové metodiky, nové techniky, když má k dispozici ten svůj branch - příkaz. Způsobuje to snad nízký stupeň znalostí RT. Programátor se sice na kursu, či přednášce dověděl, co to jsou podmínky, pravidla, činnosti a já nevím co ještě, závislé podmínky, úplnost RT a takové věci, ale nenaučil se je konstruovat prakticky. Zkonstruoval třeba v tom kursu jednu RT, na příklad: když pracovník má absenci, akce je: nedostane prémie. Když nemá, dostane prémie. A on tedy napíše: měl absenci Y N a pod tím napíše: dostal prémie, v jednom sloupci udělá X a teď si řekne: "Propána, proč jsem toto psal?,Vždyť to zvládnu jednodušším příkazem if." Není tedy přesvědčený o výhodách této metodiky. I když to bude RT o dvou řádcích, tak to zvládne složeným výrokem if: vkládá if do dalších if-then-else příkazů a už to má. A navíc ještě mnohý překladač o ničem víc nepřesvědčuje než, aby dokázal, že RT není nic jiného, než lépe nebo jinak hnízděný ("nestovaný") příkaz. Takový překladač má řadu nepříjemností. Překládá spravidla jen jednoduchým formátem omezené nebo binární tabulky. Zpravidla u mnohých překladačů chybí else pravidlo. Nemá iniciální pravidlo. Všechny podmínky v okamžiku vstupu do tabulky musí být definované, neboť překladač překládá po pravidlech. Dále nemá žádnou kontrolu logiky - nekontroluje úplnost, jednoznačnost, závislé podmínky, iracionální pravidla, neodstraňuje je, netestuje žádnou možnost zacyklování se v tabulce, má omezený formát činností.... a programátor najednou zjistí, že když napíše ten problém v Cobolu složeným if příkazem, že to dokonce má jednodušší, neboť když on teď napíše RT, v podstatě nestovaný if příkaz, tak najednou zjistí, že tam může napsat pouze PERFORM nějakou akci a o tři strany dále napíše příslušné jméno paragrafu, kde napíše, co já vím, MOVE 0 TO I. A tak místo toho, aby měl program přehlednější, má ho ještě komplikovanější. Neboť pořád musí obracet o tři stránky dále, kde performují jednotlivé činnosti. Dále - další důvod, proč je ta metoda příliš nepřesvědčivá, spočívá v tom, že překladač

je zpravidla příliš neefektivní, na příklad, že neoptimalizuje anebo generuje nákladný kód. To a další důvody, to jsou čistě [illegible] programátorské důvody, že analytik neodevzdal analýzu RT, tak on, který má vyšší třídu, vyšší plat než já, když on to nedělá, proč bych se měl obtěžovat já, proč bych to dělal? Další důvod je na příklad ten, že v kursu Cobolu se vysvětlí podrobně GO TO příkaz, vysvětlí se IF příkaz, vysvětlí se PERFORM příkaz a tak dále, ale na decision příkazy už ve školení nezbude čas. A ten, kdo se tohle kursu účastní, je celý šťastný, že vůbec zvládl techniku, jak se rozhodovat v programu a když třeba půlhodinu porozuměl někdo, kdo přemýšlel o tom příkazu decision, použití to bude jedině [illegible] dovnitř a dovnitř ven, neboť proč by si ještě komplikoval život.

Abych shrnul to, co jsem zatím řekl; jestliže programovací jazyk neobsahuje RT, běžný programátor tuto techniku nezvládne, nenaučí se v RT uvažovat. RT sice mohou být pro něj zajímavé, ale praktický význam pro něj nemají. A i když jazyk obsahuje RT, jejich širšímu používání může překážet řada věcí: konzervativnost, nízká úroveň výuky, špatný formát RT v jazyku, nechuť k analytikovi, neefektivnost překladače a pod.

Myslím si, že je třeba vyhnout překladače, které mají vzpomenuté vady, to je jedna věc. Dále kursy jazyků dělat takovým přístupem, který nejprve podrobně probere decision a potom teprve se zmíní o tom, že je to možné napsat pomocí příkazů IF, aby si zvykl dělat rozhodnutí v Cobolu pomocí decision a ne IF příkazů. Dále ten kurs by měl být výjimečně praktický, aby naučil lidi přímo konstruovat RT a hlavně by je měl přesvědčit o výhodách RT, aby byli vnitřně přesvědčeni, že RT je algoritmický jazyk stejně dobrý a vzápětí říkám, že ještě lepší než na př. PL/I nebo Cobol nebo vývojový diagram. Lepší nejen proto, že je stručnější, přehlednější, ale především proto, a na to se často zapomíná, že je to neprocedurální

jazyk. Že jedna HT je ekvivalentní na příklad nevím kolika vývojovým diagramům, podle počtu podmínek a pod., a ten překladač má šanci z těch možných všech ekvivalentních diagramů vybrat ten optimální, ten nejlepší. Neboli: já nemám šanci vybrat z tří miliard vývojových diagramů ten nejlepší, ale překladač tu šanci má. Myšlení člověka je funkční, člověk uvažuje v kategoriích toho, čemu v tabulkách říkáme pravidla. V závislosti na této skupině podmínek se provede tato skupina činností. Člověk normálně neuvažuje proceduríně. To, že my jsme museli uvažovat proceduríně, to je náš ústupek myšlení stroje - stroj myslí proceduríně. A když se nám podaří lidi přesvědčit, že HT je neproceduríní jazyk, když si to oni uvědomí, tak v tom začnou algoritmizovat. O tom budu vzápětí hovořit, jak to bylo u nás. A to je jedna věc, že je třeba lidi přesvědčit, že je to algoritmický jazyk, a ještě, když bude překladač mít inteligentní pravidla a takovéto věci, dále, že je neproceduríní, neboli lehčeji se v něm algoritmizuje, a třetí velká výhoda, o které se v literatuře vůbec nehovoří je, že je to jediný z jazyků nebo algoritmizačních prostředků, který dnes učinil velký krok dopředu v tom, čemu se říká programové dokazování správnosti programů (artificial intelligence). Jak zatím my dokazujeme správnost programu? Vyrobíme si ladicí data, program přes ně přeženeme a když výsledek odpovídá nějakému vypočítanému výsledku, tak ten program je správný. Je snaha programově dokazovat, podobně jako se dokazují na př. věty v matematice, správnost programu. Že a překladač HT má faktickou šanci dokazovat nejen syntaktickou správnost napsaného programu, to znamená odhalovat chyby kódování, chyby programátora, ale odhalovat i chyby logiky, zjistit, jestli tam máte všechny větve, zjistit, zda se vám algoritmus zacyklí nebo ne - jde o nekonečnou smyčku - a řadu věcí. Tedy překladač může kontrolovat i logiku programu, to znamená, že může zjišťovat chyby myšlení, chyby algoritmizace, chyby analýzy, programátorské analýzy. Že a když přesvědčíme člověka, že je to algoritmický jazyk, tedy neproceduríní, tedy pro člověka vhodný a že překladač kontroluje nejen

chyby kódování, ale že kontroluje navíc i chyby myšlení, tak se začne používat. My si tím kvůli u nás v Železničních asi dva roky na IBM 370 a v celku se dá říci, že po obrovském počátečním nadšení, kdy každý nedělal nic jiného, než že hned začal psát všechno násilím v tabulkách, přišla zdravá skepse. Bylo to způsobeno jednak nevhodností použití této metody násilím, na problémy, pro které se vůbec nehodí, a potom tím, že příslušný překladač nebyl optimální, generoval příliš nákladný kód. A pak se zjistilo, že na příklad asi ve třech případech použití tabulky zachránilo zpracování, tak že: algoritmus byl popsaný v zadání rozhodovací tabulkou, kterou napsal analytik, bezmezně nadmyslel a překladač na to vygeneroval větev else, do které vygeneroval zastavení výpočtu. Abnormálně ukončil program a vytiskl oznámení, že se došlo do větve else, která nebyla v původním zadání uvážena. Tedy analytik nadmyslel všechny možnosti, překladač vygeneroval větev else, která zastavila program a zachránilo to jeden dost důležitý základní soubor. Byla to chyba analytika a překladač ji zlikvidoval. A tak po té skepsi přišla jakási vyrovnání a používá se to asi v deseti, dvaceti procentech programů. Mohlo by to být více, ale my jsme asi před půl rokem po jiných dvanácti přešli z PL/I na Cobol. A ten překladač byl pro PL/I. Tak dnes to procento značně poklesalo, stále není takové jako u PL/I programů, ale programátoři často uvažují, algoritmizují v RT a potom to ručně přeloží do Cobolu, do příkazů if. Neboť se přesvědčili o výhodách RT. RT nejsou náhradou příkazů if, jsou něčím více.

U nás máme dobré počítačové vybavení a je tam i dobrý překladač RT. Velice jednoduše se ohlašuje, programátorům je to všechno dostupné, mají všechny návody k tomu, jak to používat a přes to bych řekl, že využití RT je nízké. Přesto, že třeba u nás, v našem Výpočetním centru není nám šíje velký propagátor RT, Vašek Chvátalovský. A příčiny? Já bych řekl skutečně, že: nerozumí analogii RT a nedocenění metody právě pomocí RT se ztratí velmi mnoho. Je to skutečně ten komplex

příčin, o kterých se tady mluví.

Metoda práce RT přináší bezesporu výhody. O tom jsme tady přesvědčeni asi všichni, a přesto její nízké využití je velký problém. Já si myslím, že to je v nedocenění znalostí o RT, v povrchních znalostech a vlastně pořád ještě v počátečním ostychu pracovníků, kteří s nimi nezkusili pracovat. My jsme udělali jednu velkou zkušenost, a sice, že jestliže analytik nám formuluje problém pomocí RT, tak programátor to vždycky přijme a provede program pomocí RT. Nikdy se neuchýlí k tomu, že analytické RT překóduje do vývojových diagramů. To jsme ještě nezažili. A obráceně se to ale děje, že analyticky formulované nějaké předpisy pomocí vývojových diagramů programátoři si sami kódují v RT. Já bych řekl, že větší neznalost je na straně analytiků. I větší konservatismus.

My jsme spolu s Vaškem Chvalovským udělali metodu – my jsme o tom také napsali jeden článek už asi ve 72. roce do MAA – metodu, která je vhodná pro řízení více souborů, které vstupují do programu. Když do jednoho programu vstupuje více souborů, které jsou shodně setříděny a mají se, jak tomu říkáme dohánět, srovnávají se jejich klíče a nastávají tam různé situace. Víme, že vždycky tam existuje 2^n-1 situací, přičemž n je počet vstupních souborů, a ty situace se dají krásně vyjádřit RT. Ta tabulka se nemusí ani překládat nějakým překladačem, tam skutečně stačí, když programátor prostě napíše nějaké příkazy perform, napíše si ty činnosti prostě v tom pořadí, v jakém mají být, a je z toho dobrý strukturovaný program. A to je na tom to nejcennější. My jsme poznali, že u těch programů, kde dodržíme formální strukturu, provedeme kontrolu, a i když ta kontrola neprobíhá automaticky, je snadná vyzkoušitelnost činností rovněž. Já si myslím, že u problémů, který jsme takto navrhli řešit, by měly být RT vždycky používány.

No a potom můj názor na školení programátorů. My jsme u nás v ČKD Praha poznali, že z těch vědomostí na nováčka, začátečníka je příliš mnoho. Že kurz programátorů by musel

trvat několik měsíců. Proto jsme si mysleli, že by bylo lepší, kdyby se dělalo školení programátorů dvoustupňové. Jedno to základní, kde by programátor získal základní znalosti a prostě zapadl by už do pracovního kolektivu a konkrétně by dostal nějakou práci. Další etapa jeho odborného růstu by měla být po určité, třeba půlroční až roční praxi jakási vyšší škola programování. Tam by se měly probírat jako nutná součást programátorských znalostí ve větší hloubce i RT. Možná, že je to polemické, co tady navrhuje dr. Zelenka, aby se okamžitě, předem školily RT. Nevím, jestli po těch našich začátečnících nechceme hned příliš mnoho. Ale je to otázka možná spíše pedagogická.

RT je možno vysvětlit bez návaznosti na počítač, a to je možné udělat předtím, než se dělá jakékoliv školení pro počítač. Prostě napřed vysvětlit celou metodiku RT a potom říci, jak se to dá udělat na počítači. Proto tak jak to vidím u nás, tak jsou to vlastně problémy dva. U analytiků bych řekl, že je velmi s podivem, proč ti to nechtějí používat, protože se domnívám, že třeba na úrovni analýzy je to dobrý komunikační prostředek i s tím zadavatelem, protože je to zápis poměrně přehledný, takže i ten zadavatel tomu rozumí. A je to vlastně takový spojovací článek, který on potom jenom může vlastně rozpracovat a předat programátorovi přímo k překódování. Ovšem u těch programátorů, tam je to s tím poněkud problematické, protože tak jak se u nás k tomu staví, tak mi řeknou, no když to má tři podmínky, tak k tomu nepotřebuji dělat RT, protože na to si nemusím dělat ani blokové schéma. To napíšu prostě jenom sestavou IFů a ono mi to vyjde. A samozřejmě, když je to potom něco složitějšího, tak se narazí na to, že chybí nějaký kontrolní prostředek, kterým by se dalo zjistit, zda ta tabulka je logicky správná. U nás třeba žádný kompilátor RT není. A u větších tabulek je opravdu velmi těžké zjistit, jestli ta tabulka není vlastně protismyslná.

Zkusil jsem popsat situaci okolo RT pomocí RT. Podmínky dávají nějakou diagnózu, dávají odpověď na 1. otázku seminá-ře, proč tedy se RT přes ty proklamované přednosti, proč se jich nepoužívá a ty akce potom dávají nějakou terapii - od-pověď na tu třetí otázku. Tu druhou otázku, zda tedy použí-vat metody RT, tam je odpověď celkem zřejmá. Já si myslím, že to, jestli konkrétní člověk, programátor, používá RT, zá-leží především na tom, jestli tuto metodu zvládl. Ne a jestli vůbec ví, že existuje, protože hodně je takových lidí, kteří to nevědí. Dále na tom, jestli ji aktivně umí používat. Dále to záleží na tom, jestli této metody používat může, především, jestli má kompilátor. Protože jestli ho nemá, je dost těžké přesvědčovat lidi o tom, že této metody používat mají. Potom se dost předností ztrácí. Dále to závisí na tom, jestli té tabulky používat chce, to znamená na jeho nějakých subjektiv-ních předpokladech, na jeho stanovisku. Cílem by mělo být, aby co nejvíce programátorů spadlo do prvního pravidla.

RT	Situace v používání metody RT											
if ZNÍ	Y	N	N	Y	Y	N	Y	Y	Y	N	N	
if MŮŽE	Y	Y	Y	Y	N	N	Y	Y	N	Y	N	
if CHCE	Y	Y	Y	N	Y	Y	Y	N	N	N	N	
if UŽÍVÁ	Y	N	Y	Y	N	N	N	N	N	N	N	
NÁSLEDUJ	X											
RAUČ		X				X				X	X	
POMOC					X	X			X		X	
MÁ SMŮLU				X								
PŘESVĚDČ				X				X	X	X	X	
DEJ PRÁCI							X					
VYMKNI SE			X									
CHYBA RT												X
	44%	11%					6%	39%				

Cílem našeho snažení by mělo být, aby frekvence pravidel v této tabulce zleva doprava klesaly a aby co nejvíce dotazovaných prošlo prvním pravidlem. Pod podmínkou ZNÍ v tabulce rozumíme jednak to, že dotyčný o této metodě ví a že ji umí použít. Spojili jsme obě tyto podmínky do jedné kvůli přehlednosti tabulky. Dále, použijeme-li terminologie docenta Hořejše, představují podmínky této tabulky diagnózu daného problému, akce jako terapii.

Dělal jsem pomocí toho také nějakou anketu u nás ve středisku a dopadlo to tak, že 44% lidí metodu zná, může ji používat, chce ji používat a používá ji. 11% je takových, co tu metodu nezná, používat ji může a chce, ale nepoužívá, protože ji nezná. (Mladí programátoři, co nastoupili asi před týdnem. Vědí, že existuje tato metoda, zatím jsme to nestihli je to naučit). Potom je tam jedna takové kuriózní možnost, že je tam programátor, který tu metodu zná, může ji používat, chce ji používat, ale nepoužívá. Takže se domnívám, že takovému člověku je třeba přidělit nějakou práci, protože nemá co dělat. To je jediná možná terapie. Potom je tam 39% lidí, kteří tuto metodu znají, mohou ji používat a nechtějí ji používat, což je tedy poměrně dost v našem středisku, je to skoro tolik, jako těch, co ji používají. Nechtějí ji používat a taky ji nepoužívají. U těch lidí jsem si jako asi napsal, že je třeba je přesvědčovat. Jakým způsobem je přesvědčovat, na to existuje několik možností. Na př.zmůžobnout pro tuto metodu vedoucí pracovníky a potom, aby honorovali jednotlivé pracovníky podle používání HT a na třeba podle počtu odvedených normohodin. To je celkem utopická metoda. Taky snad existují jiné způsoby, jak přesvědčit lidi o správnosti této metody. Mělo by to být především záležitostí systémového programátora. A tam by měl v tomto směru na ně působit. Brzy asi spadne naše středisko do té kategorie, že všichni programátoři metodu budou znát, budou ji chtít používat a nebudou ji umět používat. Protože dostaneme nový počítač R 4030 a tam ten kompilátor pravděpodobně nebude.

Ještě k té třetí otázce, co dělat pro to, aby se zlepšila situace v otázce používání RT. Jestli by nebylo vhodné třeba někoho podplatit, aby rozpoutal třeba v tisku nějakou útočnou kampaň proti RT, aby se to dostalo do širší programátorské veřejnosti. Třeba, když se bude vést nějaká polemika, tak ti lidé se budou zajímat o to, co to vlastně jsou ty RT. Je třeba někoho podplatit, aby se k tomu propůjčil. Další užitečná věc v oblasti používání RT, by bylo vydání nějaké normy ohledně RT. Zatím je taková situace, že u každého počítače existuje v lepším případě nějaký kompilát o RT. To je ale tak různé, že otázku nějakého přenosu RT z jednoho počítače na druhý je takřka nemyslitelná. A že vydáním nějaké té normy by se tomu pomohlo. Potom myslím, že by stála za úvahu možnost vytvoření nějakého takového portabilního kompilátoru, který by byl řekněme na bázi Cobolu, a byl by dělán totální metodou, což znamená, že po určitou co nejnižší úroveň by to bylo psáno v Cobolu, až řekněme nějaké ty podprogramy by si potom každý udělal sám. Od které úrovně by si to začal dělat sám, záleželo by od toho jednotlivého počítače. Prostě, aby to bylo psáno v nějakém vyšším programovacím jazyce. Protože jakmile se přejde na nějaký assembler, tak ta portabilita se stává dost iluzorní. Potom taky se víc přimlouvám za to, aby se víc propagovala metoda RT mezi analytiky. Jakmile se začnou tabulky používat už v etapě analýzy, potom už je jejich přechod do etapy programování poměrně snadný. Tedy větší dělí bych vynakládal mezi analytiky, kteří jsou, jak už většina tady řekla, v těchto věcech více konzervativní.

Na rozdíl od vás jsem právě analytik a sám za sebe chci říci, že souhlasím s názorem, že RT by v prvé řadě měli používat právě analytici. A naše zkušenosti s tím:

Zatím se RT u nás nepoužívají. U nás máme důsledně oddělenou analýzu od programování. Už řadu let jsou tyto činnosti vyhraněny - my si zásadně písemnou cestou předáváme zadání programů. To je myslím důležité říci, protože praxe je v řadě VS různá. Pokud jsou tyto profese spojeny, tak i přístup k RT

bude asi jiný. My jsme s RT začali asi tak před půldruhým rokem, kdy já v analýze jsem si řekl, že by bylo záhodno v rámci vylepšování stylu práce se začít RT zabývat. Z vlastní iniciativy a s podporou vedoucího odboru jsem se po tom začal pídit. Něco jsem si o tom přečetl, navštívil jsem taky některá VS, posbíral tam o tom nějaké zkušenosti a potom jsme se domluvili s programátory, že v tom něco uděláme. Že oni vyvinou nějaký překladač pro naše dva počítače, no a to se také stalo. V květnu letošního roku jsem udělal pro analýzu o metodě RT přednášku - tedy půldenní školení, kterého se zúčastnila většina analytiků. Zpracoval jsem také takový standard k RT. Smyslem toho bylo, aby se hned se začátku začalo používat nějakých jednotných termínů v oblasti RT a i grafická forma byla také jednotná a podobná. K tomu školení jsem udělal také nějaké příklady - školské i konkrétní specifikace programů. To dostali do všech analytických oddělení. Nicméně nechalo se to na dobrovolnosti. Řekli se: "Teď máte školení, příklady, překladač a zkuste si to udělat. Uvidíte jestli se vám to bude hodit nebo ne." Uplynulo půl roku a zjišťujeme, že skutek utek', že se skutečně objevuje to, co tedy bylo řečeno řadou z vás, že jen školení asi nestačí. A že pokud si to někdo skutečně v praxi nezkusí, nenaučí se tabulky sestavovat, jinak myslet než byl doposud zvyklý. Prostě představa, že to někdo sám od sebe po takovém školení začne používat se ukázala pofidérní. A tak jsme se dohodli - když nezabrala dobrovolnost, no tak je asi třeba i nějaký donucovací prostředek - vyhlásit třeba nějakou soutěž v počtu RT a stimulovat to i finančně, případně i s nimrnými postihy - tedy kriteria jak pro řadové pracovníky, tak i specialisty a vedoucí oddělení, kteří jsou kromě jiného zodpovědní za to, aby v rámci svého oddělení i celého odboru nové metody a nové poznatky propagovali a šli příkladem. Projednávalo se to na poradě odboru. No a teď byl oheň na střeše. Argumentovalo se různými způsoby, převážně asi v tom smyslu, že v analýze máme teď fůru mnohem vážnějších problémů, které by se měly řešit,

se kterými bychom se měli vypořádat a my se tady tak přehnaně důkladně zabýváme RT, které zhruba v naší práci tvoří možná šestinu objemu, pokud tedy máme na mysli tvorbu specifikací programů. Protože je skutečně pravda, že s dnešními ASŘ a podobnými vzletnými koncepcemi se většinou vyšíváme v takových nadoblačných výškách a potom směrem k tomu programování, k vlastní specifikaci programů - sejména analytici na vyšších funkcích - se velmi zřídka dostáváme. To je možná takových deset, dvacet procent naší práce. Je tu ovšem zase řada dalších pracovníků, taková polovina odboru bych řekl, jejichž práce zase převažuje právě ve specifikaci programů.

Tedy proč se RT zabýváme tak moc, když nás pálí řada dalších problémů, a mnohem závažnějších? Protože si myslím, že nějakým způsobem by se měla prosadit každá nová a dobrá věc. V opatřeních jsem navrhoval rozdělit celou realizaci do dvou etap. První by měla charakter čistě školicí, aby se s RT skutečně každý pracovník naučil dělat, aby k tomu prakticky přičichl, protože to školení, kde mě každý snad celkem pozorně vyslechl, ale nic víc, to nestačilo. No a k tomu směřovalo právě i ta soutěž v počtu RT ve specifikacích a jiných projekčních materiálech, i ty finanční motivy. Oponovalo se - my tady budeme násilně plodit řadu RT jen proto, abych vyhrál nějakou tu stovku, ačkoliv problém mohu popsat či znázornit třeba jednodušším způsobem. Já osobně tvrdím, že i to má svůj smysl - tedy v tomto případě po dobu té první, školicí etapy (předpokládal jsem tak jeden rok). Zkoušet a konstruovat tabulky třeba snad i na ne zrovna vhodném místě, i na triviální rozhodovací situace, které bych třeba popsal programátorovi jednodušeji slovně. Znovu opakuji, že smyslem by bylo, aby se člověk naučil RT používat a konstruovat a získal jakýsi cit pro vystižení situace, kdy je či není účelné RT použít či použít jiného vyjadřovacího prostředku. No a potom jak to budou lidé ovládat, pak nastupuje už druhá etapa realizace, kdy by se tabulky začaly používat opravdu jenom v těch vhod-

ných případech, tam, kde je to účelné, tedy optimálně. Samozřejmě, že v této fázi by už nějaké soutěžení v počtu RT a podobně bylo nesmyslné a zrušilo by se.

Pokud se týká toho vhodného a účelného použití RT. U nás je totiž situace poněkud komplikovanější tím, že máme už několik let důsledně zavedené normalizované programování. My analytici jsme také podle jednoho našeho standardu povinni zadávat programátorům specifikace programů podle zásad tohoto normalizovaného programování. Tím se podstatně oproti dřívější praxi už dost napomohlo lepšímu vyjádření funkce programu a zestručnění jeho specifikace. Dříve si to každý specifikoval, jak se mu zlíbilo, každý měl prostě svůj přístup a styl. Většinou se popisovalo slovně a v některém takovém slovním popisu složitějšího problému se stěží kdo vyznal - bylo to tak zamlžené, že to člověk četl dvakrát, třikrát a nebyl z toho vůbec chytrý. Ale zavedením normalizovaného programování se program rozsekal na bloky, každý ví, co ve kterém bloku má popsat a vyjádřit a tak ta komunikace mezi programátory a námi je za ty dva tři roky dost vylepšená a vypěstovaná. Takže tady potom nastává další problém, jak teď do tohoto popisu programu v duchu normalizovaného programování vhodně vměstnat ještě RT. Nesmí totiž dojít k tomu, aby obě metody se vzájemně potíraly, aby jedna narušovala zásady a výhody té druhé. V zásadě si myslím asi toto: Analytik by se neměl snažit (alespoň na v té prvé fázi, v prvním období, než se dospěje k nějakým zkušenostem a závěrům) pomocí RT se pokoušet předepisovat programátorovi, jak to má programovat. Aby skutečně řešil jen věcnou stránku problému, který chce pomocí RT popsat - tedy bez ohledu na to, jak si to pak programátor naprogramuje. Prostě aby dělal RT analytické. K tomu ještě pro dokreslení naší situace: Dnes náš analytik se stále více vzdaluje programátorovi. Jak už jsem řekl, dochází tu k hlubší diferenciaci s titulu koncepčních prací na ASŘ a i rostoucích nároků na zvládnutí všech detailů programování (nové metody, prostředky a pod.). Analytik dnes je mnem více

tíhnout k uživatelům a ke zvládnutí věcné stránky řešených problémů. Diferenciace je už někdy značná a dochází se k názoru, že i v rámci analýzy se budeme a už začínáme specializovat - asi ve smyslu koncepčních pracovníků na jedné straně a pak vlastních projektantů systémů na počítač na straně druhé. Tím chci tedy říci asi to, že analytik i z tohoto důvodu není ve většině případů schopen kápnout s RT programátorovi plně do noty a udělat tabulku tak, aby on ji mohl vzít a prostě přepsat a dát ji ke strávení přímo překladači. Máme zato, že asi tu bude muset nutně být období, kdy programátor si musí zvážit, jestli analytikovu RT si předělá na RT programovou, tedy "podle obrazu svého", nebo to dokonce naprogramuje jiným způsobem. Snaha ovšem je, aby překladač byl využíván, čili aby si programátor vytvářel případně i sám RT z RT analytických.

Takže asi takový je teď momentální situace u nás - jsme právě v tom stavu, kdy prostředky máme, ale hledáme způsoby, jak překonat konservatismus a někdy i nesnášenlivost. Hodně lidí bude tvrdit, že je to nevhodné, že je to pro ně zbytečné, že máme mnohem závažnější problémy, ale žádný z těch, kteří takto mluví, ještě ani jednu tu RT neudělal. A to je, myslím, hlavní kámen úrazu - i když na druhé straně nutno konstatovat fakt, že se na každého snáší tolik práce a úkolů, že mnohdy není čas, ani chuť se učit něco nového.

Ano, to věřím, není čas ani chuť se učit něco nového. A najde to uspěchat. Podle mého názoru se nedá říct - uplynulo půl roku a skutek utek. Půl roku je málo! Víme, že jiné progresivní metody se leckde prosazovaly řadu let. A prosadily se.

Způsoby, jak se progresivní metody prosazují do praxe, jsou různé. Chtěl bych ukázat na analogii mezi metodou RT a metodou hodnotové analýzy. Ta má taky řadu výhod a taky proniká pomalu do praxe. Někde se postupuje tak, že někdo absol-

vuje školení, avšak vysvětlení dostane to, ve, až použije u některého řešeného problému metodu hodnoto analýzy. Nepostupovat stejně u MT?

Já jsem jako perlička, když jsem dělal na dvou přesvědčovacích akcích pro používání MT, tak mi jedna ten analytik řekl, že teďka nemají čas přemýšlet, že musejí programovat.

Problém, jak přesvědčovat, jak přimlouvat lidi, je dost obtížná položená otázka. Já bych řekl, že to je otázka publicity nebo prostě zveřejňování výsledků. Literatury kolem MT už je, ale na mne působí takovým nesympatickým dojmem. Tam jsou samé přednosti, tam není jediná nevýhoda, ono je tak sice nasnadě, že to není všelék naprosto pro všechno, ale v podstatě se všude publikují úspěchy. To je všecko pravda, ale člověk tak nějak vnadu tuší, že je tam nějaké "ale". To "ale" nikdo neřekne, a to působí aspoň na mne velmi deprimujícím dojmem, že si říkám to asi tam bude nějaký žertovo kopýtko.

Další věc: myslím si, že třeba ve vašich podmínkách určitě se někde najde, kdo s tím začne pracovat, s těmi MT. Pokud se budou nějakým způsobem publikovat ty jeho výsledky, ať už úspěchy nebo neúspěchy, dojde k infiltraci.

Dále problém optimalizace. Na Tesle by takový optimalizující překladač musel jet hodiny, aby vytvořil s MT optimální program.

To je právě ten rozpor: na počítačích menších, pomalejších, s omezenými zdroji, tam by ta optimalizace byla nejvíc zapotřebí, a tam je nejhůře proveditelná. Na výkonných počítačích je více možné MT překládat optimálně, ale je to tam málo důležité.

Tam snad je spíš ten psychologický dojem, že pokud je možná ta optimalizace, tak to má tu výhodu, že nemůže přijít programátor a říct: vždyť ten kompilátor je vlastně špatný,

vždyť já bych to napsal líp. I to může mít nějaký význam pro to, že se raději rozhodnu pro tu tabulku, protože mám zajištěné, že to budu mít dobře.

Využiji výhody, kterou mám před všemi vámi, že mluvím tady poslední, když už jsem vyslechl celou řadu názorů a dovolím si tedy ve svém vystoupení se dotknout některých věcí, které tady už přišly na přetřes. Především bych chtěl odbýt jednu věc: Ta se týká metodiky výuky jazyka. My jsme tady hodně mluvili o RT až na té úrovni překladače nebo dokonce i kompilátoru, to znamená kdy pro využití RT musíme mít příslušný výrok nebo výroky příslušného jazyka, byl tady uváděn příklad cobolu. Mám za to, že v žádném případě by nebylo asi na místě při výuce jazyka pomíjet to obecné a základní jádro jazyka, a tím, že dáme přednost některému rozšíření jazykem speciální aplikace. Já se domnívám, že zásada, jestliže vyučuji některému programovacímu jazyku, je, že se v první řadě musím držet toho obecného jádra, a tím, že mohu případně přidat to, co je vhodné v rámci jednotlivých počítačů nebo lépe řečeno v rámci jednotlivých kompilátorů.

Pokud jde o problematiku samotnou, tak se domnívám, že sešlo jsme se zaměřili příliš specializovaně, příliš na tu čistě programátorský pohled, a to tak specializovaně, že dokonce se tady nesly názory programátorské, tedy RT dobré, pokud existuje překladač, pokud neexistuje překladač - význam podružný a tak. Dovolím si tady mít poněkud odlišný názor, já vám to hned odůvodním. A ještě prosím vás, k tím překladačům samotným byly tady vysloveny kritiky některých překladačů, mluvilo se tady o optimálním překladu a tak tedy - já bych chtěl v zásadě říct asi tohle: pokud jde o tu optimalizaci překladu, tak tedy je známo, že existují dnesko metody, které dovolují tu optimalizaci provádět, ovšem optimalizace se děje s ohledem na dvě hlediska. Můžeme provádět optimalizaci s ohledem na délku vygenerovaného kódu, a můžeme provádět optimalizaci s ohledem na efektivnost při zpracování programu. A právě

ta dvě hlediska do jisté míry stojí v protikladu, to znamená, že prakticky asi každý překladač tabulek bude volit nějakou takovou střední cestu mezi těmito dvěma hledisky, takže my nebudeme moci říct, že budeme mít optimální program. Nebo je možné (existují taky takové přístupy) volit optimalizaci překladu z jednoho nebo druhého hlediska. Ale mám za to, že provedení té dokonalé optimalizace vyžaduje tak náročnou práci toho překladače, že ten praktický výsledek nebývá vždycky úměrný té ceně, kterou za to musíme platit. A ta cena, kterou za to musíme platit, ta spočívá u některých překladačů v tom, že dovolují překládat pouze tabulky omezeného typu, na př. tabulky s omezenými vstupy, popřípadě, že jsou málo flexibilní.

K samotné tematice: domnívám se, že jsme to pojali příliš úzce. Z toho důvodu, že RT jsou daleko univerzálnější prostředek než jenom prostředek programování, a i když náš seminář v Havířově je zaměřen na metody programování, a tedy i na toto programátorské použití RT, tak se nebudeme moci v žádném případě vyhnout té problematice, která má styčné body s pracovníky, kteří nám program zadávají, a nebudeme se moci vyhnout té úloze, kterou RT má kromě toho, že je vyjadřovací prostředek, a to jako prostředek dokumentační. A v souvislosti s tím bych chtěl říct toto asi: RT, jak každý z nás ví, může být zapsána na velmi různých úrovních: může ji používat analytik při vyjasňování problému v té části své práce, kdy diskutuje zadání se zákazníkem, může ji využívat jako takového vnitřního komunikačního prostředku mezi tím, kdo tedy řekněme tvoří koncepci systému a tím, kdo bude navrhovat jednotlivé programy, no a bude pochopitelně RT také součástí programového zadání a programové dokumentace. A já se domnívám, že jeden zdroj potíží a averze vůči RT ze strany toho zadávajícího, v daném případě jsem mluvil o analytikovi, že spočívá v tom, že není vyjasněno, do jaké podrobnosti má analytik tu tabulku rozepisovat. Já jsem slyšel přednášku Ing. Kutáče a tam uváděl příklad, kdy tedy je program řešen RT, které jsou rozepsány až do úrovně někde úplné jednotlivých výroků, tedy prakticky RT,

která je napsána tak, že ji mohu zadat k prostému překódování. Nebo možná, když bych měl překladač, že tu tabulku mohu převést, když tedy podmínky a činnosti vyjádřím cobolskými výroky, že ji budu moci přímo předložit překladači. Ale já si myslím, že pro nás bude ohromným přínosem, jestliže budu mít RT ve specifikaci programu, kde podmínky a činnosti budou popsány ne na úrovni jazyka, tedy programových výroků, ale kde budou popsány na úrovni věcné problematiky. To znamená, že mně ta RT přinejmenším ten program rozčlení do naprosto jasných logických celků, kdy takový jeden celek budu pojímat jako jednu činnost v rámci této RT, a tím dosáhnu nesmírného zlepšení při zadávání toho programu. Protože já takto popsaný program mám připraven k tomu, abych ho mohl řešit třeba - modulárně, třeba jinými metodami progresivními, strukturované nebo v rámci normalizovaného programování a tak. Myslím si, že vyjasnění těchto okolností by možná mohlo hodně přispět. Svérázná situace a ovšem situace obtížnější, nastává tam, kde kromě RT, případně jejich překladačů, jsou k dispozici ještě jiné prostředky, tedy automatizované prostředky jako pomůcky programování. Je to situace našeho výpočetního střediska. My máme k dispozici kromě překladače RT také generátor normalizovaného programování. A pochopitelně, že chceme-li používat obou těchto věcí, tak musíme oba tyhlety prostředky nějakým způsobem sladit dohromady. Není na př. v naší situaci použitelná ta cesta, o které mluvil Ing. Ševčík z ČKD, to jest, kdybychom řešili, tak, jak má třeba Chvalovský ve své knize popsáno, řízení programů z několika vstupních souborů, kdybychom řešili RT, tak bychom ten generátor mohli prostě hodit do koše. Takže v našem případě bude specifický problém, který spočívá v tom, používat RT tak, aby řešila problematiku pouze v rámci jednoho bloku toho normalizovaného programování. Pak se tyto dvě metody dají obě sloučit dohromady a tedy nedostávají se do konfliktu.

Nakonec bych se chtěl vyjádřit ještě k těm určitým psychologickým bariérám, na které narazíme při tom zavádění. Já

myslím, že nepředěláme psychologii člověka natolik, když pominem nejmladší generaci, tak jsme všichni zatížení dost velkou dávkou konzervatismu. Já se domnívám, že dokonce i různé donucovací metody tady mohou být dost málo efektivní. Domnívám se, že přesvědčovací metoda by měla být tedy prioritní. A mohla by existovat velice pádná přesvědčovací metoda. Ovšem bylo by potřeba udělat jednu věc. Vzít nějaký složitější program, který je ve specifikaci popsán donucovací metodou, to znamená, kdy třeba převažuje slovní popis. To by byl program, kde jsou třeba dvě, tři desítky stránek slovního popisu, když pominu třeba popis vstupních a výstupních dat, myslím jen popis algoritmu. A vyjádřit na problémové úrovni tohleto v RT. A pochopitelně, příslušně taky sprogramovat. Tedy tu tabulku tam začlenit do programu příslušně spracovaného pomocí překladače. A myslím si, že kdybychom předložili ke srovnání tuhletu klasickou specifikaci s programem z ní zakódovaným a specifikaci, která obsahuje RT, s programem podle toho udělaným, že by každý mohl dostat do ruky velice přesvědčivý dokument o tom, jak RT může být výhodná.

Totiž to je ta technika smetany. Předložit problém, který je ideální pro RT. Já si myslím, že stejně účelné by bylo říct: třeba tento problém je výhodný pro RT, ale tento, tam se ten efekt už ztrácí - že je to neutrální a tento je pro to nevhodný. Tímto postupem získáme daleko víc příznivců těch RT, než že jim budeme předkládat tu smetanu - si řeknou- ano, to je ta smetana a co je pod tím. Každá věc má dvě strany.

Mně nebude vadit, když mně někdo ve specifikaci napíše primitivní RT. Protože mne nikdo nemutí, abych tuto primitivní RT, kterou mohu vyjádřit jedním výrokem, překládal. Já budu překládat takové tabulky, které stojí za ten překlad. Jestliže je mohu popsat v rámci jednoho výroku, tak na to nebudu tahat překladač.

Ještě k překladači. Již tady bylo řečeno, že samotná existence dobrého překladače nestačí k dostatečnému využití RT. Můj názor je, že překladač je užitečná nadstavba, nikoliv nezbytně nutná.

Zajisté, RT přinášejí dost užitku, aniž máme prostředek k jejich automatickému překladu. Ovšem na druhé straně si myslím, že ideální je, jestliže ten prostředek máme. Ať každý, bez ohledu na počítač a operační systém, má možnost skusit RT se vším všudy, i s tím automatickým překladem. Bylo by třeba vyvinout jeden překladač RT, který by byl portabilní, t.j. snadno přenositelný na různé počítače. Zatím se zdá, že to je jenom utopie.

Já myslím, že by to zase taková utopie nebyla. Neboť jestliže na napsání použijeme třeba cobol, a to takovou podmnožinu cobolu, která obsahuje všechna omezení, tak až na pár modulů by se to dalo napsat.

Již samotná koncepce zpracování RT předpřekladačem kompilátoru vyššího programovacího jazyka nabízí možnost přenositelnosti překladače RT, neboť překladač je vázán především na hostitelský jazyk, a tedy by mělo být možno jej přenášet na různé počítače, které jsou vybaveny kompilátorem hostitelského jazyka. Tento požadavek by byl splněn na příklad tehdy, kdyby překladač RT byl sám napsán v hostitelském jazyce, a to v takové jeho podmnožině, kterou umí zpracovat (a v níž se nijak neliší) kompilátory hostitelského jazyka na jednotlivých počítačích.

Domnívám se, že jazyk cobol by mohl poskytnout takovou podmnožinu, která by byla společná aspoň počítačům EC, včetně systému OS, a možná i pro počítač TESLA. Změny v překladači pro jednotlivé počítače by se asi nemusely týkat jiných částí než ENVIRONMENT DIVISION. Jazyk cobol navíc poskytuje dosti účinný přeložený program, takže obava z malé efektivnosti

překladače napsaného v cobolu není příliš opodstatněná.

Já bych tu obavu formuloval poněkud jinak. Bylo tady řečeno, jaké ten překladač má mít vlastnosti, aby splnil své poslání. A jsou to náročné požadavky. Když to shrneme, vidíme, že to bude složitý systémový program, rozhodně nic triviálního. A tam na příklad cobol nebo i jiný hostitelský jazyk nemusí být ten pravý prostředek. Nemusí jít o efektivnost, ale přímo o proveditelnost. Na příklad v tom překladači pracujeme s vnitřními datovými strukturami typu seznamů. Nebo sama metoda postupného rozkladu vede velmi přirozeně k rekurzi. Myslím, že části by měly být na př. v assembleru.

Jaká je moje představa? Představuji si publikaci, obsahující perfektní, odladěnou specifikaci takového překladače. Předpokládám, že jde o velký, modulární celek. Publikace by měla obsahovat specifikaci, funkční specifikaci všech částí, všech modulů. Ať si každý, kdo to chce zkusit, implementuje překladač sám, ale s podstatně menším úsilím, než kdyby měl promýšlet i návrh. Důležitá je důvěra v předlohu, vědomí, že podle této specifikace již byl implementován fungující program pro jiný počítač, jiný operační systém. Jsem přesvědčen, že je možné napsat specifikaci nezávislou na počítači nebo operačním systému. Samozřejmě, budou-li v té publikaci některé části již také naprogramovány, třeba i většina, dojde tomu v tom cobolu, tím lépe.

Takže nakonec chcete totéž, co bylo řečeno předtím. Jenomže navíc požadujete, aby se takový portabilní produkt předával s úplnou a dobrou dokumentací, jejíž součástí je aktualizovaná specifikace. Což je rozumný požadavek. V každém případě je tady tam požadavek portability, požadavek nezávislosti na typu počítače nebo operačního systému. A to je úsilí navíc. Ať už jde o psaní specifikace nebo o omezení na podmnožinu hostitelského jazyka. Příčemž přínos tohoto úsilí navíc nebude ani tak pro podnik jako celospolečenský. Takže,

pokud nepůjde třeba o státní úkol nebo o něco podobného, je tady problém, kdo to udělá. A ještě něco. Aby se takové věc podařila, je nutné udělat určitý první krok. A ten by možná byl v silách někoho z nás, třeba aspoň částečně. Tím nutným prvým krokem myslím podívat se na pravidla zápisu RT, umožněná některými překladači, na dosavadní standardizační snahy a standardizovat jazyk RT, třeba pro ten cobol.

Závěr: K prvé otázce, proč není metoda RT dosud dostatečně používána, je v diskusi několikráte konstatována nedostatečná znalost a konservatismus. Převládl názor, že překladač je velmi užitečná nadstavba, i když podle mínění některých účastníků nikoli nezbytně nutná. Ukázalo se, že samotná existence dobrého překladače nestačí k dostatečnému využití RT.

Ke druhé otázce, zda je masové používání RT žádoucí, odpověděli všichni účastníci kladně.

Nejzávažnější je ovšem třetí otázka, co dělat. Propagovat, popularizovat a školit. Ovšem tak, aby se školení nemínilo účinkem. Přes určité rozpory v názorech na způsob školení můžeme shrnout především požadavek na vysokou kvalitu školení ("dobrá hodinová přednáška je lepší než špatné nebo žádné"), na řadu vyřešených příkladů z praxe, na to, aby školení bylo dálkové nebo vícestupňové. Za znovuvyslovení stojí i názor, že by snad bylo dobré, aby kurs RT byl zaměřen i na strukturované a normalizované programování. Většina diskutujících zdůrazňuje, že přesvědčovací akce by měla být zaměřena především na analytiky. Při propagování RT je dobré si uvědomit jejich silné stránky. Zasvěcený popularizátor apeluje hlavně na vlastnosti vyplývající z funkčního, neproceduálního vyjádření problému pomocí RT.

Je nutno ocenit i negativní poznatky potvrzující, že přesvědčování a řádné školení je vhodnější než domlouvání.

Bylo řečeno, že je také třeba vybavit výpočtová střediska

našich uživatelů účinnými překladači. Náš původní úmysl, dát zde zájemcům stručný přehled překladačů dostupných v ČSSR, chceme z objektivních důvodů uskutečnit později. V diskusi se ukázalo, že ještě žije myšlenka portabilního překladače, snad se jednou stane skutkem.

Řada námětů z této diskuse bude znovu, podrobněji rozpracována ve speciálních referátech. Pozorný čtenář si jistě všiml několikrát vysloveného návrhu, že by bylo seriozní ukázat i nevýhody RT, ukázat, kde se ta metoda nehodí. Nalézt slabou stránku metody RT mi zpočátku připadalo jako úkol nesmírně obtížný, téměř nadlidský, nicméně zkušenější kolega se toho v jednom z dalších referátů dotkne. Také nejasnosti kolem použití metody RT a jiných progresivních metod, strukturovaného programování, normalizovaného programování, pokud nebyly beze zbytku vyjasněny zde, najdou ještě místo na stránkách tohoto sborníku. A zvláště atraktivní možnost automatického ladění logiky programu, věc, kterou snad docenímě teprve zítra. Také se uskutečnil návrh z diskuse, aby se jeden referát zabýval standardizací jazyka RT. A také k nejzávažnější problematice, k problematice výuky, se ještě vrátí jeden referát.

Přehled literatury o RT

je sestaven tak, aby obsahoval velké množství českých a slovenských pramenů. Pokud jde o cizojazyčnou literaturu, uvádíme nevelký počet pramenů. V uvedených pramenech mohou houževnatí zájemci nalézt odkazy na další rozsáhlou zahraniční bibliografii.

1. DŽUPPOVÁ,G.: Použitie RT v systémovej analýze; Ekonomika stavebníctva, IX. č.4, 1974.
2. EISHOLZ,G.: Datenstrukturierung, Entscheidungstabellen und Verbund /Spojitost řízení souborů, RT a dokumentace/. Büretechnik 1973, č.1.
3. GREVE,R.: Entscheidungstabellen als systematisches Hilfsmittel in der ADV. /RT jako systematický pomocný prostředek v automatizovaném zpracování dat/. Büretechnik

1973, č.1. Zde seznam další literatury.

4. HAVRDA,J.: Algoritmus překladu RT na počítači EC 1021; Používání jazyka COBOL, sborník přednášek: ČsVTS Dům techniky Pardubice, červen 1975.

5. HAVRDA,J. KRATOCHVÍL,E.: PROTAB-Překladač RT pro počítač EC 1021; dtto.

6. HAVRDA,J. CHLOUBA,J. KRATOCHVÍL,E.: PROTAB-překladač RT pro počítač EC 1021 - uživatelská příručka: VÚMS Praha IX 1977.

7. HAVRDA,J. CHLOUBA,J. KRATOCHVÍL,E.: PROTAB-překladač RT pro počítač EC 1021 - programátorská příručka: VÚMS Praha, IX 1977.

8. HAVRDA,J. CHLOUBA,J. CHROUSTOVSKÝ,J. KRATOCHVÍL,E.: RT - metoda racionalizace projektování a programování; sborník přednášek k semináři v Českém Krumlově; ČSVTS, komitét pro vědecké řízení Praha 1978.

9. HUMBY,E.: Programy na základě RT; ALFA, Bratislava 1977.

10. CHVALOVSKÝ,V.: Rozhodovací tabulky; SNTL Praha 1974.

11. CHVALOVSKÝ,V.: Použití RT při vývoji software; INORGA č. 1, 1971.

12. CHVALOVSKÝ,V.: Rozhodovací tabulky; sborník Výpočetní technika ČKD, Praha 1970.

13. CHVALOVSKÝ,V.: Automatické vytváření programů z RT; Inorga č.3 1973.

14. CHVALOVSKÝ,V.: Aplikacia RT; Informačné syatémy č.2. 1973.

15. KEŠNER,J.: Rozhodovací tabulky. Český komitét pro vědecké řízení, Praha 1972. Zde seznam další literatury.

16. KEŠNER,J.: Úvod do RT; INORGA Praha 1970.

17. KEŠNER,J.: Rozhodovací tabulky v praxi; Podniková organizace č.7 1971.

18. KEŠNER,J.: Blokové schemata nebo RT? INORGA č.1 1971.

19. KEŠNER,J.: RT a blokové schemata; sborník:Podnikové řídící syatémy, Dům techniky ČsVTS Praha 1973.

20. KEŠNER,J.: RT a jejich využití v budování ASŘ; VÝBĚR č.3, 4,5,6 1976.

21. KOHYDA,F.: RT-metóda zobrazenia logických rozhodnutí; Informačné syatémy č. 1 1972.

22. KRATOCHVÍL,E.: Co jsou RT? NEVIO č. 6, 1972.

23. KRATOCHVÍL,E.: Automatická konverse RT; MAA č.11, 1974.

24. KRATOCHVÍL,E.: PROTAB-překladač RT pro počítač EC 1021, MAA č. 10, 1975.

25. KRATOCHVÍL,E.: Automatické spracování RT počítačem, Aktuality výpočetní techniky, č. 4, 1974.

26. KRATOCHVÍL,E. SEDLÁŘ,M.: Metoda RT v řízení; studijní texty; Institut ČsKVR Praha 1974.

27. KRATOCHVÍL,E. SEDLÁŘ,M.: RT, podstata, význam, aplikace; Podniková organizace č. 7, 1973.

28. KRATOCHVÍL,E. SEDLÁŘ,M.: Využití RT v řídící praxi; Podniková organizace č. 9-10, 1975.

29. KRATOCHVÍL,E. SEDLÁŘ,M.: Rozhodovací tabulky; Aktuality výpočetní techniky č. 4, 1973.

30. KRATOCHVÍL,E. MIROVSKÝ,M. ŠPROGL,J. PRŮŠA,J.: Rozhodovací tabulky; VÚŘA Praha 1970.

31. KRATOCHVÍL,E. SEDLÁŘ,M. LAMPERT,M.: Pětijazyčný výkladový slovník základních pojmů metody RT; Podniková organizace č. 1, 3, 4, 5 1976.

32. KRATOCHVÍL,E. HAVRDA,J.: Překlad RT na počítači EC 1021; Aktuality výpočetní techniky č. 2 1976.

33. LACKO,B.: Rozhodovací tabulky; MAA č. 9 1970.

34. LÁCKO,B.: Používání RT v praxi analytika a programátora; Informačné systémy č. 4, 1973.

35. LACKO,B.: Používání rozhodovacích tabulek; Sborník přednášek Metody programování počítačů III.gen.Havířov 1975

36. LAMPERT,M.: Využitie RT pre kontroly a korekcie; programové zabezpečenie vstupného modulu ISIS, VVS-OSH Bratislava 1973.

37. LAMPERT,M.: Spracovanie RT a ich použitie; dtto.

38. MATTER,L.: Entscheidungstabellen in ADV und Fachbereich /RT v autom.zprac.dat a v odborných útvarech/. Bürotechnik, 1976, č.3.

39. PETERS,H.: Entscheidungstabellen als Organisationsmittel in Sachbearbeitung und EDV /RT jako organizační prostředek při řešení věcných problémů v automatizovaném zpracování dat/. Bürotechnik, 1973, č.11.

40. SAUER,S.: Entscheidungstabellen - Aufbau und Einsatz /RT, jejich sestavování a využívání/. Bürotechnik, 1973,č.9.

41. SCHUPP,W. HAAS,P.: Strukturierte Programmentwicklung durch Entscheidungstabellen-Technik und normierte Programmierung /Strukturované programování pomocí RT a normovaného programování/. Bürotechnik, 1973, č.11.

42. SEDLÁŘ,M.: Speciální aplikace RT; Automatizace č. 7 1972.

43. SEDLÁŘ,M.: Aplikační vývoj metody RT; Informačné systémy č.1 1975.

44. SUCHOMEL,E.: Přínosy RT pro řízení rozhodovacích procesů; Podniková organizace č. 5, 1972.

45. ŠEVČÍK,B. CHVALOVSKÝ,V.: Mocninová metoda řízení více souborů v programu; MAA č. 6, 1972.

46. VLČEK,J.: Rozhodovací tabulky; Ekonomicko-matematický obzor č. 4, 1967.

47. BEIGANG,O.: Entscheidungstabellen - Vorzüge und Anwendungsgrenzen bei der Programmgestaltung /RT - přednosti a meze při zpracování programů/. Bürotechnik, 1973, č. 1.

48. HOFFMANN,K.: Automatische Umwandler für Entscheidungstabellen /Automatické překladače pro RT/. Bürotechnik, 1973, č.1. Zde rozsáhlý seznam další literatury.

49. THURNER,R.: Umwandlung von Entscheidungstabellen /Překladače RT/. Bürotechnik, 1973, č. 11. Seznam literatury.