Pavel Šimoník
OKR - AŠ k.ú.o.

ROZHODOVACÍ TABULKY JAKO SOUČÁST SPECIFIKACE A DOKUMENTACE PROGRAMU

Můj příspěvek se bude zabývat tematikou, o níž bylo již mnoho napsáno. Jestliže se přesto chystám připojit k tomu několik dalších stránek, jsem k tomu veden různými důvody. Především je pokud zarážející, že je v naší práci relativně velmi málo využíván prostředek, který je již dosti dlouho na světě a jenž se zejména v posledním desetiletí dostalo značné publicity v literatuře. Při tom existuje dnes již základně propracovaná teorie rozhodovacích tabulek, byl znalostně doplněn výběr problém automatického překladu rozhodovacích tabulek do kódu programu a také metodika aplikace rozhodovacích tabulek jak v systémové analýze, tak v programování byla již dobře propracována a publikována.

Často se spatřují příčiny tohoto velmi zdržlivého postoje k rozhodovacím tabulkám v přirozeném konzervativismu a lenosti lidí, v příliš malé pracovní zátěži, která nedovoluje rozhlédnout se lépe kolem sebe v oblasti našeho profesionálního zájmu. Tyto důvody mají nepochybně svou váhu. Mne však více zaujal jeden názor, vyslovený na jednu z přípravných pracovních schůzek našeho letošního semináře o rozhodovacích tabulkách se vždy píše jen o té dobré stránky, na mnoha příkladech se demonstrují jen jejich nepopiratelné výhody a jen tak na okraj se připomíná, že snad někdy vyjímečně by mohly být nástrojem nevhodným, nebo že by s jejich použitím mohly být spojeny nějaké problémy. Právě z toho pak prorostá přirozená nedůvěra, jakou máme nakonec ke všemu, čemu se dělá nepřiměřená reklama.

Pokusím se tedy učinit takto položenou otázku východiskem dalších úvah a pojednat o rozhodovacích tabulkách jako o jednom z vhodných pomůcek, které mohou posloužit k tomuto cíli, t.j. k přesné specifikaci programu a k jeho dobré dokumentaci. Při tom se ukáže, že samotná metodika aplikace není dána jen nástrojem, kterého bodláme používat, ale též dalšími okolnostmi. Takový to pohled nám umožní alespoň nastínit cesta k řešení některých problémů, s nimiž se při aplikaci nepochybně setkáme.

Problematiku osvětlíme na příkladě programu a úkolu, který byl nakonec specifikován pomocí rozhodovacích tabulek s cílem využít při zakódování překladače rozhodovacích tabulek ve spojitosti s generátorem normalizovaného programování. Specifikace je pochopitelně zjednodušena v tom smyslu, že rozpis vět souborů s nimiž program pracuje, je uveden ve zkrácené formě pouze s ohledem na potřeby tohoto jediného programu. Na druhé straně je ve specifikaci alespoň velmi stručně naznačena věcná problematika, aby čtenář získal o programu lepší představu.

Než přistoupíme k rozboru příkladu, považuji za nutné vyjasnit některé otázky týkající se specifikace programu a také cíle, které si klademe při využití rozhodovacích tabulek ve specifikaci. Není pochyb o tom, že specifikace musí obsahovat přesný popis vstupů a výstupů se všemi informacemi, které jsou k vybudování programu nutné. Dále musí nezbytně obsahovat přesný popis algoritmu řešení úlohy. Tuto věc je možno podat ve dvojím podstatně různém provedení. V jednom případě bude v sobě popis algoritmu zahrnovat pouze popis logiky programu, nikoliv však způsob technického realizování řešení úlohy v rámci programu. Při tomto pojetí vede na programátorovi, aby pro vybudování programu použil nejvhodnější programovací techniku a také všechny různé prostředky, které k tomu má k dispozici. Při tomto přístupu k věci se programátor významně podílí na tvorbě programu, při které může být s výhodou využito jeho specifických znalostí. Jde tedy přirozeně na mysli programátora-analytika. V druhém případě je

možno ve specifikaci zadat přímo programovou realizaci algoritmu řešení a takto specifikovaný program předat k pouhému zakódování ve zvoleném jazyce. Při tomto pojetí bude nezbytně nutné, aby se programátor-analytik podstatnou měrou podílel na vypracování specifikace, protože pro systémového inženýra s jeho širokým záběrem přístupu k věcné i speciální problematice by bylo sotva únosné udržovat se na výši současného stavu programovacích metod, i kdyby sám praktické programování ovládal. Ve spojitosti s těmito dvěma přístupy ke tvorbě specifikací bude i rozhodovací tabulka ve specifikaci mít poněkud jiné postavení. V prvém případě bude hrát především významnou úlohu dokonalého komunikačního prostředku, bude však nutno ji zpravidla ve větší nebo menší míře změnit, jestliže chceme využívat překladače a tabulku převzít do zdrojového programu. Tím přirozeně poklesne dokumentární hodnota tabulky ve specifikaci. Ve druhém případě získáme dobrou korespondenci mezi tabulkou ve specifikaci a tabulkou v programu, ale v přesném popisu jednotlivých činností se může ztratit globální pohled na řešený problém a na postup řešení. Na připojeném příkladě se může čtenář názorně přesvědčit, že bez krátkého slovního vysvětlení by ze samotných tabulek vnikl do podstaty popsaného řešení jen s určitými potížemi.

Obraťme se nyní k našemu příkladu. Poněvadž jsme si již řekli, že jsme stanovili za cíl využít rozhodovacích tabulek v souvislosti s využitím generátoru normalizovaného programování, je pro nás nepřijatelná cesta, při které bychom celou specifikaci algoritmu vyjadřovali výlučně rozhodovacími tabulkami, ačkoliv tato metoda je v literatuře některými autory hojně propagována a je jinak jistě zásadně možná. Naše rozhodovací tabulky mají tedy zůstat omezeny pouze na rámec jednoho bloku v normalizovaném programu a nemohou navíc obsahovat činnosti, které jsou řešeny v rámci generátoru. Vidíme tedy, že obsah rozhodovacích tabulek ovlivňují současně použité programovací prostředky, zatím co jejich forma je ovlivněna pojetím specifikace.

Při sestavování tabulek v našem programu jsme se snažili zůstat ve specifikaci na úrovni analytických, t.j. problémově a nikoliv programově orientovaných tabulek, je však

zřejmé, že se to zdaleka nepodařilo. V tabulkách jsou sice některé podmínky a činnosti vyjádřeny slovně a názorně, ale vyskytují se tam také činnosti, jako je manipulace s daty v pracovní paměti, obsazování výhybek, testování výhybek a pod., což samozřejmě do čistě problémově orientované tabulky nepatří. Příčinou tohoto odbočení od zamýšlené cesty byly potíže s nalezením přesné slovní formulace podmínek a činností, která by adekvátně vyjadřovala řešený problém a při tom byla natolik stručná, aby se dala zapsat do tabulky. Nakonec jsme se uchýlili k jisté celkem přirozené symbolice, kterou je možno považovat za praktický návrh, netroufám si však říci, nakolik zdařilý. Tyto problémy spojené s vyjadřováním v rámci tabulky jsou po mém soudu jednou z příčin malé obliby rozhodovacích tabulek u nás. Při specifikaci tabulkou jednodušších dílčích problémů se tyto potíže podstatně zredukují nebo odpadnou vůbec, při specifikaci složitějších úloh je však s nimi nutno počítat.

Rozdělení popisu zpracování znakové věty do tabulek RT-Z1, RT-Z2 bylo provedeno tak, aby pokud možno zůstala v jedné tabulce pravidla s určitými společnými skupinami činností. Rozdělení na tabulky menšího rozsahu by nebylo účelné, ponévadž by si jednak vyžádalo více psaní, jednak by vedlo ke ztrátě komplexního přehledu všech relevantních podmínek.

Naproti tomu zpracování kmenové věty bylo nutno rozdělit do čtyřech tabulek z toho důvodu, že zde dochází k testování řady na sobě nezávislých podmínek a mnoha možnými kombinacemi stavů. Dovolím si vyslovit z hlediska našeho tematu kacířskou, ale po mém soudu snela opodstatněnou myšlenku: Kdybych neměl k dispozici překladač rozhodovacích tabulek, dal bych pro ruční zakódování tohoto úseku programu přednost vývojovému diagramu. Zásadu používat dané situaci nejpřiměřenější prostředek by se měla vždy dodržovat.

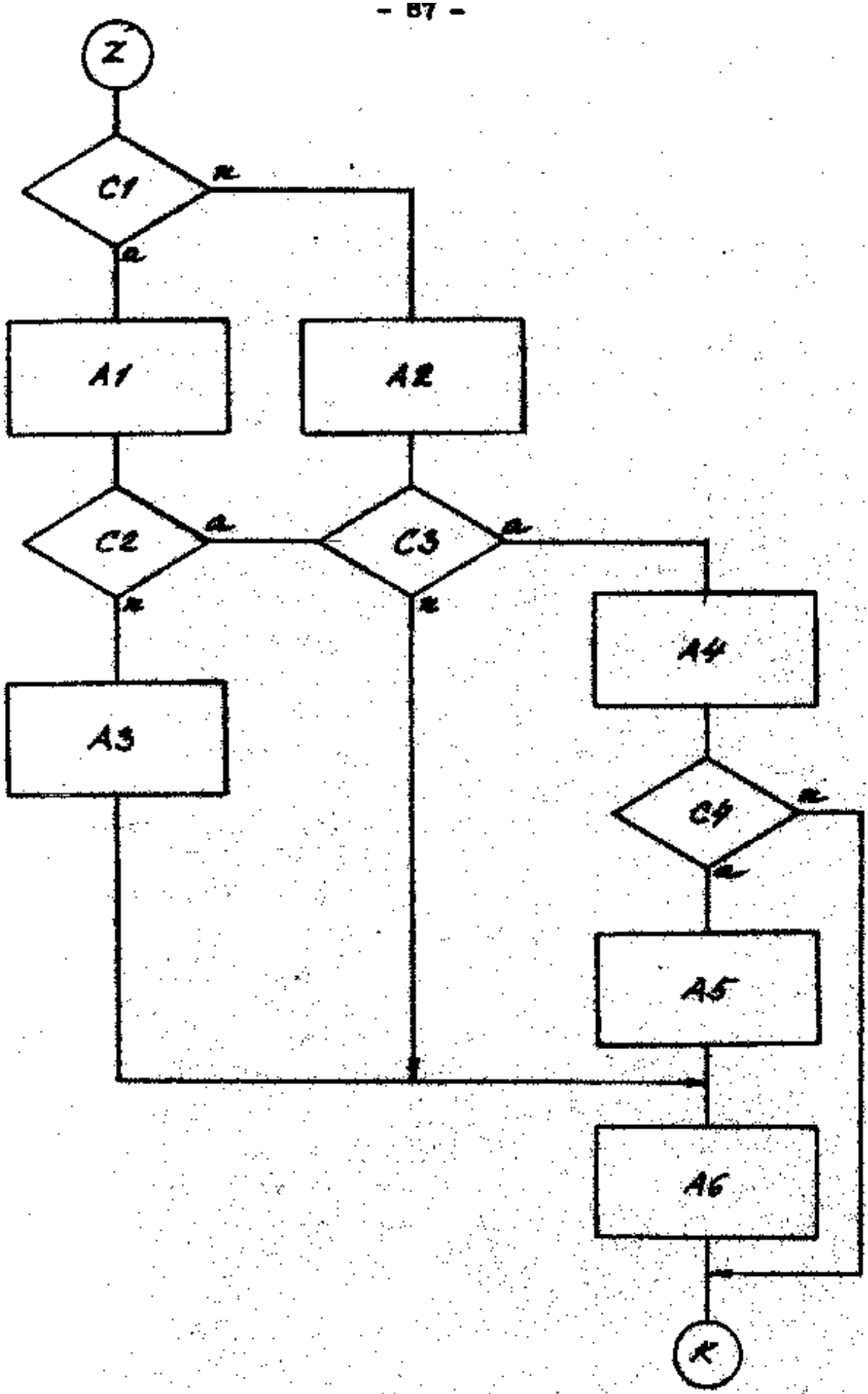
Při vytváření programových tabulek ze zadaných tabulek analytických v naší specifikaci dojde k několika změnám. V prvé řadě musají být jinak vyjádřeny slovní formulace některých podmínek a činností. Dále si všimněme, že se jak v tabulkách pro zpracování znakové věty, tak v tabulkách prro zpracování kmenové věty se často vyskytuje činnost "vytvoř větu 095 z kmen. souboru", což je z hlediska efektivnosti na-

vhodné a bude dobře zařadit tuto činnost mimo tabulky bezprostředně po přečtení kmenové věty.

Ze všeho, co jsem dosud uvedl, vyplývá tento závěr: Pokud je na pracovišti taková organizace práce, že jsou od sebe oddělany útvary programování a systémové analyzy, je nezbytné, aby nejen koncepce programu, ale i forma zadání byly vzájemně konsultovány, aby u systémového analytika nevznikal dojem, že je na něj přenášena část práce, kterou dosud neměl ve své pracovní náplni. Pokud možno, měly by být rozhodovací tabulky dodávané analytikem ve specifikaci orientovány čistě problémově.

Zatím jsem mluvil o využití rozhodovacích tabulek v daném programu dosti kriticky a snažil jsem se nezatajovat žádnou z potíží. Musím však na druhé straně říci, že rozhodovací tabulky se v tomto případě projevily jako nástroj nadmíru užitečný. Program byl totiž původně specifikován bez použití rozhodovacích tabulek v jiné jednodušší verzi. Když k němu byly dodána znalma se slovním popisem algoritmu aktualizace, ukázalo se, že ve specifikaci jsou temná místa, která se podařilo vyjasnit a algoritmus precisně formulovat teprve za použití techniky rozhodovacích tabulek při objasňování problému. Tak vznikla tato verze specifikace daného programu, a to je zvalín pro aplikaci rozhodovacích tabulek nejlepším doporučením.

Nakonec bych se rád zmínil o jednom argumentu, který bývá často uplatňován proti použití rozhodovacích tabulek. Tvrdívá se že mnoho, ne-li většina programů z oblasti zpracování ekonomických informací má převážně lineární průběh, který se pouze jednoduše větví, a to na různých místech. Tato námitka by byla závažná, kdyby takové koncepce programu vždy nutně vyplývala z jeho logiky. Je tomu tak ve skutečnosti? Všimněme si daného programu, který je popsán vývojovým diagramem na obrázku 1. Podmínky C2, C3 a C4 mohou mít takovou povahu, že jejich testování je skutečně možné až po předchozím provedení některých operací v předcházejících blocích A1, A2, A4. V takovém případě je hořejší námitka plně oprávněná. Vyjadřovat testování shodných dvou podmínek a příslušné rozvětvení programu v místě, kde na sebe navazují testy podmínek C2, C3



Obr. 1

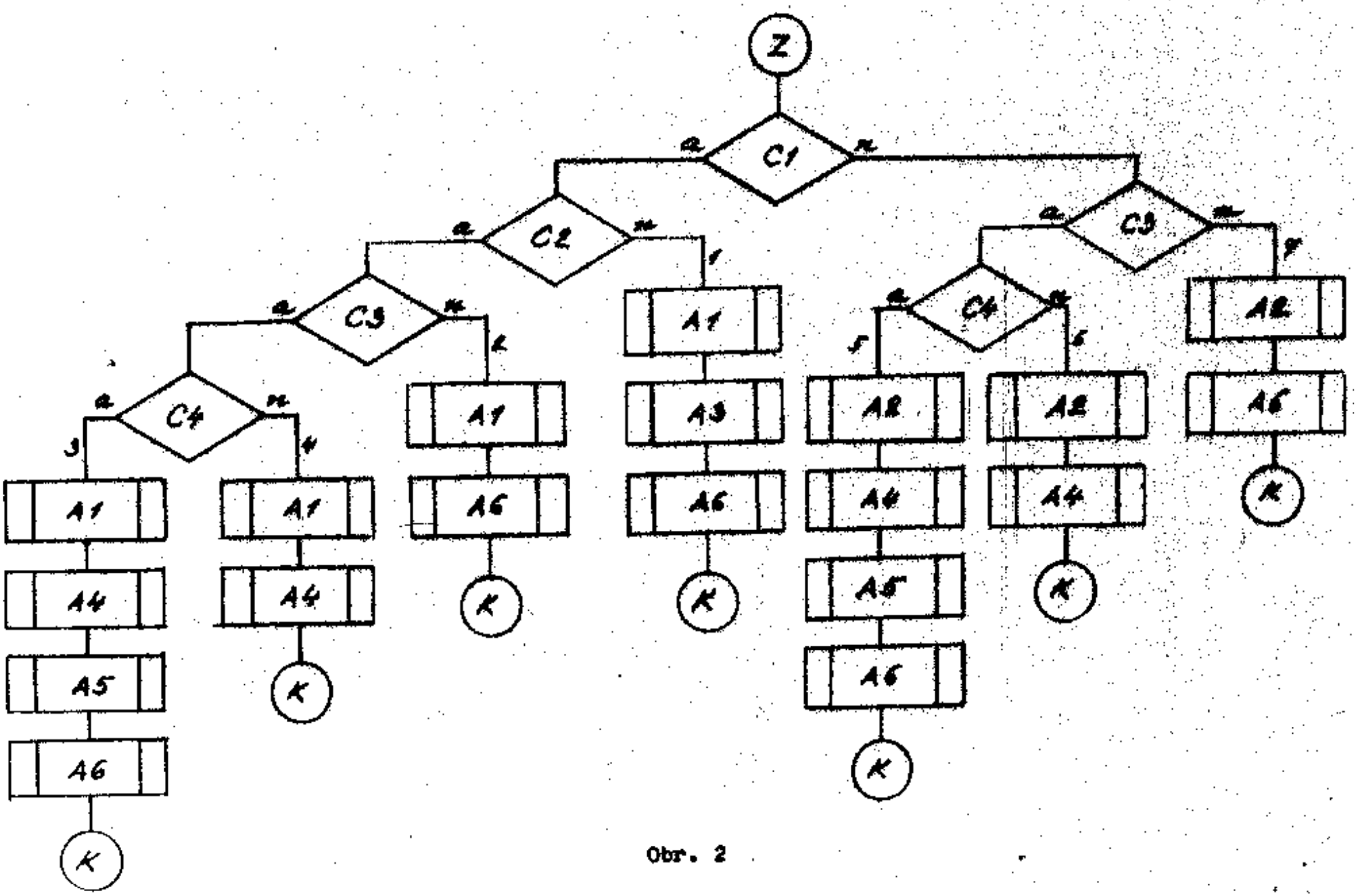
rozhodovací tabulkou by bylo nepřiměřené. Mnohem jednodušeji splní ten účel výrok typu

```
IF C2
   IF C3  ....
   ELSE   ....
ELSE  ....
```

Ale co když mají podmínky C1, C2, C3, C4 takovou povahu, že bychom je mohli všechny testovat hned na začátku popsaného úseku programu? Pak se ukáže, že vývojový diagram na obr. 1 není důsledkem logiky programu, ale našeho návyku lineárního myšlení. Použijeme-li místo vývojového diagramu ve specifikaci ekvivalentního slovního popisu, podaří se nám tento stav ještě lépe zamaskovat. V takovém případě by se však již vytvoření rozhodovací tabulky vyplatilo. Vývojovému diagramu na obr. 1 odpovídá na př. tato rozhodovací tabulka:

	1	2	3	4	5	6	7	8
C1	Y	Y	Y	Y	N	N	N	
C2	N	Y	Y	Y	-	-	-	
C3	-	N	Y	Y	Y	Y	N	
C4	-	-	Y	N	Y	N	-	
A1	X	X	X	X				
A2					X	X	X	
A3	X							
A4			X	X	X	X		
A5			X		X			
A6	X	X	X		X		X	

Tuto tabulku můžeme jednodušeji vyjádřit jiným vývojovým diagramem (viz obr. 2!). Porovnáme-li oba diagramy, vidíme jasně oč v tomto případě jde. Využívání rozhodovacích tabulek ve specifikacích nás přirozenou cestou přivádí k jinému způsobu myšlení, což je sice zpočátku třeba obtížné, ale přináší s sebou velmi cenný "vedlejší efekt" - naučíme se modulární myslet a také koncipovat modulární programy, takže postupem času dosáhneme i v konkrétních programech stavu, který odpovídá současné úrovni programovacích technik.



Obr. 2

Specifikace programu FA44.

Stručná charakteristika programu:

Program aktualisuje věty kmenového souboru FA431 konstrukčních prvků (dále jen KP) údaji z vět souhlasně setříděného změnového souboru FA401. Opravené kmenové věty zapisuje na výstupní soubor FA441. Současně vytváří výst. soubor FA432, který obsahuje jednak věty chyb, jednak věty s údaji o vyřazených konstrukčních prvcích.

Popis vstupů, výstupů a algoritmu aktualisace podle zásad normalisovaného programování:

B. Vstup:

FA401 - Změny základních údajů pro aktualisaci KP.
Soubor obsahuje věty 061 a má poř. číslo 1.

FA431 - Kmenový soubor KP.
Soubor obsahuje věty 400, 490, 500 shodné struktury a má pořadové číslo 2.

Oba soubory jsou vzestupně setříděny podle klíčů
1. Základní číslo pasportu
2. Druh pasportu
3. Pořadové číslo pasportu.

Struktura souboru FA431:
Pro každé základní číslo pasportu existuje jako prvá v pořadí vždy buď věta s druhem pasportu 1, za níž následují věty s druhem pasportu ≠ 1 nebo 9, nebo věta s druhem pasportu 9 a poř. číslem pasportu 0, za níž následují věty s druhem pasportu 9 a poř. číslem pasportu ≠ 0.

Výstup:

FA441 - Doplněný kmenový soubor KP.
FA432 - Soubor chyb a vyřazených KP.
Soubor obsahuje věty 095 chyb a věty 491 vyřazených KP.

D. Činnosti při změně klíčů.

Při změně základního čísla pasportu nuluj oblast W1 v pracovní paměti (viz dále).

E. Algoritmus aktualizace.

1. Změnovými větami se aktualizují kmenové věty se shodnými hodnotami klíčů 1 až 3, a to postupně v pořadí přicházejících změnových vět. Pro každou hodnotu klíčů 1 až 3 existuje jediná věta na kmenovém souboru (bez ohledu na typ). Při tom se zároveň vytvářejí věty 095 nebo 492. Tato aktualizace včetně případu, kdy ke změnové větě neexistuje odpovídající věta kmenová je podrobně popsána rozhodovacími tabulkami RT-21, RT-22.

2. Jestliže dojde v kmenovém souboru ke změnám ve větách s druhem pasportu 1 nebo s druhem pasportu 9 a pořadovým číslem pasportu 0 (pasport domu), je třeba zachytit v pracovní paměti stav příslušných opravovaných údajů. K tomu účelu je třeba v pracovní paměti definovat oblast W1 s položkami závod, bytový obvod a datum vyhotovení. Kromě toho bude oblast W1 obsahovat pole s 6 jednobytovými položkami znak změny. Obsazení těchto položek hodnotou 1 (podle druhu prováděné změny) poslouží k záznamu, zda a které věty byly při aktualizaci pasportu domu zapisovány na soubor TA3 chyb a vyhovujících KP. Tyto činnosti jsou rovněž definovány v RT-21 a RT-22.

3. U každého jiného pasportu téhož základního čísla je nutno po provedení aktualizace podle bodu 1 opravit ještě příslušné údaje hodnotami z oblasti W1 (pokud není nulová) a rovněž tak provést zápis vět chyb a vyhovujících KP obdobně, jako při aktualizaci pasportu domu. [illegible] to znamená, že změny v pasportu domu se promítnou rovněž do příslušných pasportů bytů, [illegible] nebo [illegible]. Tyto činnosti jsou prováděny při zpracování kmenové věty a jsou definovány v RT-41, RT-42, RT-43, RT-44.

Zpracování změnové věty.

RT-Z1	1	2	3	4	5	6	7	8	9	A	B	C	D	E
Ke zm. větě ex. km. věta	N	X	X	X	X	X	X	X	X	X	X	X	X	
ZM.DRZM = ?	-	1	1	1	1	1	1	6	6	6	6	7	7	
ZM.DRPA = ?	-	1	1	9	9	-	-	1	1	9	9	1	9	
ZM.PØBC = 0	-	-	-	X	X	-	-	-	-	X	X	-	X	
KM.TIPV = 400	-	X	N	X	N	X	N	X	N	X	N	-	-	
500 → KM.TIPV			X		X		X		X					
ZM.ZAV → KM.ZAV			X	X	X	X	X	X	X	X	X			
ZM.BØ → KM.BØ			X	X	X	X	X	X	X	X	X			
ZM.PR → KM.PR			X	X	X	X						X	X	
ZM.DRZM → KM.DUVT			X	X	X	X	X	X						
ZM.ZAV → W1-ZAV			X	X	X	X			X	X	X	X		
ZM.BØ → W1-BØ			X	X	X	X			X	X	X	X		
1 → W1-ZN (ZM.DRZM)			X	X	X	X			X	X	X	X		
Vytvoř v. 095 ze zm. věty	X													
1 → DRCH	X													
Vytvoř v. 095 z km. věty			X	X	X	X								
0 → DRCH			X	X	X	X								
Zapiš větu 095	X	X	X	X	X									
Přejdi na RT-Z2														X
KØNEC	X	X	X	X	X	X	X	X	X	X	X	X	X	

Poznámka k následující tabulce:

Obsazení výhybky W1-ZRUS hodnotou 1 znamená, že při zpracování příslušné kmenové věty (podle klíčů 1 až 3) nedojde k jejímu přepisu na opravený kmenový soubor.

RT-Z2	1	2	3	4	5	6	7	8	9	A	B	X
ZM.DRZM = ?	3	3	3	4	4	4	4	4	5	5	5	
ZM.DRPA = ?	1	9	-	1	9	2	3	4	1	9	-	
ZM.PORC = 0	-	X	-	-	X	-	-	-	-	X	-	
490 → KM.TYPV	X	X	X									
ZM.DRZM → KM.DUVT	X	X	X	X	X	X						
ZM.DAT → KM.DAT	X	X	X	X	X	X						
ZM.DAT → W1-DAT	X	X		X	X							
1 → W1-ZN (ZM.DRZM)	X	X		X	X				X	X		
9 → KM.DRPA				X		X						
0 → KM.PORC				X								
Vytvoř v. 095 z km. věty	X	X	X	X	X	X	X	X	X	X	X	
7 → DRZM	X	X	X									
8 → DRZM							X	X	X	X	X	
9 → DRZM				X	X	X						
Zapiš větu 095	X	X	X	X	X	X	X	X	X	X	X	
Vytvoř a zapiš věty 491	X	X	X									
1 → KL-ZRUS							X	X	X	X	X	
KONEC	X	X	X	X	X	X	X	X	X	X	X	X

Zpracování kmenové věty.

RT-K3	1	2	3	X
KM.DRPA = 1	X	X	X	
KM.DRPA = 9	-	X	-	
KM.PORC = 0	-	X	-	
W1-ZNAKY = 0	-	-	X	
Přejdi na RT-K4	X	X	X	
Přejdi na RT-K2				X

RT-K2	1	2	3	4	5
Wl-ZN (1) = 1	X	Y	N	N	
Wl-ZN (6) = 1	-	-	X	Y	
KM.TYPV = 400	Y	N	Y	N	
500 → KM.TYPV	X		X		
Wl-ZAV → KM.ZAV	X	X	X	X	
Wl-BØ → KM.BØ	X	X	X	X	
1 → KM.DUVT	X	X			
Vytvoř v. 095 z km. věty	X	X			
0 → DRCH	X	X			
Zapiš větu 095	X	X			
Přejdi na RT-K3	X	X	X	X	X

RT-K3	1	2	3	5
Wl-ZN (3) = 1	Y	Y	N	
Wl-ZN (4) = 1	Y	N	Y	
490 → KM.TYPV	X	X		
3 → KM.DUVT	X	X*		
4 → KM.DUVT			X	
Wl-DAT ← KM.DAT	X	X	X	
9 → KM.DRPA	X		X	
Vytvoř v. 095 z km. věty	X	X	X	
9 → DRCH	X		X	
Zapiš větu 095	X		X	
7 → DRCH	X	X		
Zapiš větu 095	X	X		
Vytvoř a zapiš věty 491	X	X		
Přejdi na RT-K4	X	X	X	X

	1	2	3	4
KT-K4				
K1-ZKUS = 1	X	N	N	
W1-ZN (5) = 1	-	X	N	
KM.TIPV = 400	-	-	X	
Kz. změn. údaje ke kmen. větě	-	-	N	
0 → K1-ZKUS	X			
Vytvoř v. 095 z kmen. věty		X	X	
4 → DRCH			X	
8 → DRCH		X		
Zapiš větu 095		X	X	
Zapiš kmenovou větu			X	X
KONEC	X	X	X	X

Vytvoření věty 095

a) ze změnové věty: Přenesou se údaje základní číslo pasportu, druh pasportu a pořadové číslo pasportu. Doplní se druh chyby. Ostatní údaje jsou nulové.

b) z kmenové věty: Přenesou se údaje základní číslo pasportu, druh pasportu, pořadové číslo pasportu, závod a byt. obvod. Ostatní údaje po doplnění druhu chyby jsou nulové.

Vytvoření věty 491

Věta se vytváří pro každý konstrukční prvek vyřazeného pasportu přenosem odpovídajících si údajů z věty kmenového souboru.

Specifikace věty 061 — MP soubor FA401

Adresy od - do	Uložení	Formát	Obsah prvku	Označení
0 - 2	U	3	Typ věty	TYPV
3 - 7	U	5	Zákl. číslo pasportu	CPAZ
8	U	1	Druh pasportu	DRPA
9 - 11	U	3	Poř. číslo pasportu	PØRC
12	U	1	Druh změny	DRZM
13 - 21	U	9	Rezerva (nuly)	
22 - 23	U	2	Závod	ZAV
24 - 25	U	2	Bytový obvod	BØ
26	U	1	Prostředí	PR
27 - 30	U	4	Datum vyřazení	DAT

Specifikace věty 095 — MP soubor FA432

0 - 2	U	3	Typ věty	TYPV
3 - 7	U	5	Zákl. číslo pasportu	CPAZ
8	U	1	Druh pasportu	DRPA
9 - 11	U	3	Pořadové číslo pasp.	PZRC
12 - 13	U	2	Závod	ZAV
14 - 15	U	2	Bytový obvod	BØ
16 - 37	C	22	. . . (nuly)	
38	U	1	Druh chyby	DRCH
39 - 47	C	9	. . . (nuly)	
48 - 50	P	5,1	. . . (nula)	
51	P	1	. . . (nula)	
52 - 53	P	3,0	. . . (nula)	
54 - 56	P	5,1	. . . (nula)	
57 - 60	P	7,0	. . . (nula)	
61 - 65	P	9,0	. . . (nula)	

Specifikace věty 491 Soubor FA432

0 - 2	U	3	Typ věty	TIPV
3 - 6	U	4	Číslo KP	CKP
7	U	1	Prostředí	PR
8 - 14			. . .	USEK1
15 - 16	P	3	Počet oprav	PŠOPP
17 - 24			. . .	USEK2
25 - 26	P	2	Rok vyřazení	RŠK

Specifikace vět 400, 490, 500 Soubory FA431,FA441

0 - 2	U	3	Typ věty	TIPV
3 - 7	U	5	Základní číslo pasportu	CPAZ
8	U	1	Druh pasportu	DRPA
9 - 11	U	3	Pořadové číslo pasportu	PŠRC
12 - 13	U	2	Závod	ZAV
14 - 15	U	2	Bytový obvod	BŠ
16	U	1	Prostředí	PR
17 - 20	U	4	Datum zrušení pasportu	DAT
21	U	1	Důvod tisku	DUVT
22 - 41			. . .	
42 - 43	P	3	Čítač KP (maximálně 150)	CIT
44 - 47	U	4	Číslo KP	CKP
48 - 49			. . .	
50 - 56			. . .	USEK1
57 - 65			. . .	
66 - 67	P	3	Počet oprav (od pořízení KP)	PŠOPP
68 - 71			. . .	
72 - 79			. . .	USEK2
80 - 85			. . .	

Posledních 42 By se opakuje tolikrát, kolik KP pasport obsahuje. Maximální počet KP = 150.