U D R Ž O V A T E L N O S T P R O G R A M O V Ý C H S Y S T É M Ů

Doc.Ing.Jan Honzík,CSc

Příspěvek je přehledným zpracováním publikace /1/. Výsledky dosavadního stavu výzkumu v oblasti návrhu a výstavby programových systémů i vlastní experimentální zkušenosti vedou autory k závěrům, že použití určitých architektonických principů při návrhu a výstavbě systémů vede k dobře udržovatelným programům. Udržovatelnost úzce souvisí se spolehlivostí. Základním projevem spolehlivosti programu je jeho bezchybný chod i při změnách v programu, zádání, nebo jiných vlastností programu. Autoři nabízejí formální prostředky pro kvantitativní srovnání udržovatelnosti různých variant programu.

1. Struktura programového komplexu

V současné době jsou již dostatečně známy principy multiprogramování i současného rozpracování úkolů (multitasking). Při multiprogramovém režimu sdílí několik uživatelských programů jeden počítač. Řídicí program aktivuje a deaktivuje programy podle pravidel, která určují dalšího uživatele. Při současném rozpracování úloh může mít navíc uživatelský program několik "inkarnací"(převtělení) a řídicí program poskytuje časové úseky jednotlivým inkarnacím uživatelských programů.

Navrhované struktura je založena na přesvědčení, že takový systém sdílení času je vhodnou předlohou pro architekturu programů. Program se dělí na zcela nezávislé části a každá z nich má spojení pouze s řídicím programem. Ten plánuje provedení každé části podle potřeby. Paralelní části musí být reentrantní, protože není ekonomické vytvářet samostatnou kopii programu pro každou inkarnaci.Řada aplikací neodpovídá idealizovanému modelu pro sdílení času. Připustíme-li však dvě výjimky z pravidel výstavby, pak budou vhodná pro všechny aplikace. První výjimka umožní jistá sdílení dat mezi jednotlivými částmi a druhá dovolí jednotlivým částem komunikovat jistým, dobře definovaným způsobem, mezi sebou. Při každém návrhu zůstávají v platnosti zásady: nezávislé programové části, nejmenší možné sdílení dat a nejmenší možná vzájemná komunikace. Takový návrh již vyhovuje široké třídě aplikací; lze ho rozšířit tak, aby

vyhovoval i pro nejsložitější systémy v reálném čase nebo zúžit tak, aby byl vhodný i pro jednodušší požadavky. Většina částí se nemusí rozpracovávat paralelně, ani není nutné aktivovat části po časových úsecích. Návrh struktury vyžaduje oddělení logiky programu a programových dat. To umožní izolovat data a minimalizovat komunikaci. Návrh struktury znázorňuje obr.1. a 2. v němž se vyskytují tyto části:

Program - je souhrn objektů programové řídicí logiky a dat. Provádí funkce dané problémem. Sestává z řídicí části, z několika problémových částí a datové báze.

Třída - je soubor objektů prog.říd.logiky a je největší podčástí programu. Zahrnuje podprogramy, třídu dat a třídu modulů. Data třídy jsou dostupná jen vnitřním komponentám třídy. Existují dva typy tříd:

Řídicí třída - její podprogramy provádějí plánování a aktivační funkce programu.

Problémová třída - její podprogramy hrají roli uživatelských programů v prostředí sdílení času.

Mezi řídicí a problémovou třídou jsou pouze tyto rozdíly:

*Řídicí třída obsahuje plánovací programy pro všechny problémové třídy a je médiem pro komunikaci zpráv mezi podprogramy.

*Problémové třídy jsou aktivovány řídicí třídou a pracují v přiděleném časovém úseku.

*Řídicí třída obsahuje moduly dosažitelné z problémových tříd pro účely komunikace s říd. třídou a pro poskytování globálních funkcí.

Podprogram - je soubor prog.říd.logiky.Je největší podčástí třídy a sestává z modulu těla, z dat podprogramu a z modulů podprogramu. Říd. program aktivuje modul těla podprogramu. Moduly těla a podprogramu se mohou vzájemně volat. Žádný objekt vně podprogramu nemůže volat jeho moduly nebo získat jeho data.

Modul - je samostatně kompilovatelný soubor programové říd. logiky a je největší podčástí podprogramu. Sestává z dat modulu , řídicího segmentu a pracovních segmentů. Říd.segment je vstupním/výstupním bodem modulu.

Segment - je soubor prog.říd.logiky a je podčástí modulu. Je nejmenší části v hierarchii struktury programu. Sestává ze souboru dat a zdrojových příkazů a řeší jednu funkci. Má jeden vstupní a jeden výstupní bod. Všechny segmenty mají přístup k datům modulu.

Zpráva - je datový objekt, který se používá pro komunikaci mezi

podprogramy. Mechanismus zprávy poskytuje dvě funkce:
*Komunikace mezi podprogramy formálním, dobře ovládaným způsobem
*Dodání prostředků pro rozhodování o tom, který podprogram se má aktivovat a kdy.

2. Vztah struktury a udržovatelnosti programu

Vztah mezi popisovanou strukturou a udržovatelností programu lze stručně shrnout do několika bodů:
*Vysoký stupeň vzájemné izolovanosti jednotlivých komponent podstatně nebo zcela vylučuje nežádoucí vliv změn provedených v jednom modulu na modul druhý.
*Komunikační spojení a monitorování se provádí prostřednictvím báze zpráv. V každém okamžiku obsahuje báze zpráv záznam okamžitého stavu historie komunikace v systému.
*Určení chyby je možné výpisem báze zpráv. Většinou se prostor chyby zúží na malé množství možností, což lze využít k použití vhodných diagnostických prostředků.
*Kontrola přetížení systému se provádí monitorováním báze zpráv. Procento zaplnění báze indikuje v každém okamžiku zátěž systému.
*Navrhovaná struktura významně zjednodušuje řízení programu.
*Navrhovaná struktura zajišťuje větší průhlednost pro vícepočítačové konfigurace.

3. Stanovení udržovatelnosti

V průběhu sedmdesátých let se věnovalo značné úsilí studiu a vývoji programovacích technik. Cílem byly systémy s vyššími kvalitami, snížení času potřebného k vývoji a redukce nákladů vynaložených za celou dobu života programového systému. Na konci návrhu programu na nejabstraktnější úrovni se často dělají analýzy i zkušební simulace, které mají prověřit kvality navrhovaného systému. Žádná z metod však neodpovídá na otázky jako:
*Jak udržovatelný bude výsledný program?
*Jak adaptibilní bude celý program, nebo které jeho části budou adaptibilní?
*Jaké budou náklady na zbývající vývoj systému a kolik bude stát údržba takto navrženého systému?
*Jak lze porovnat jednotlivé varianty návrhu mezi sebou z hlediska jejich citlivosti na interakci a tudíž i na udržovatelnost a testovatelnost?

Jsou to složité otázky. Příspěvek uvádí návrh, jak lze odpovědět na tyto otázky za předpokladu, že se dodrží architektura struktury programu popsaná v předcházejících odstavcích.

Základy predikčního určení udržovatelnosti pomocí matic propojení jsou uvedeny v /2,3/. Tato metoda přináší významnější užitek v případě dodržení popisované struktury programu. V čem spočívá proncip metody predikčního určení udržovatelnosti?

*Nechť P je matice n x n, jejíž prvek i,j representuje pravděpodobnost, že změna v modulu i způsobí změnu v modulu j.

*Nechť A je matice 1 x n, jejíž prvek a_j je počet počátečních změn v modulu j.

Pak $T = A (1 + P + P^2 + P^3 + ...) = A (1 - P)^{-1}$

je matice 1 x n (vektor T) a prvek t_j představuje úhrnný počet změn v modulu j způsobený počátečními změnami v modulech, které jsou uvedeny v matici A.

Je-li možné pro každou variantu návrhu určit matici P, pak vyhodnocení vektoru T umožní zhodnotit vlastnosti návrhu a odhadnout oblasti potenciálních problémů. Jestliže např. $a_j=0$ pro j≠i a $a_j=1$ pro j=i, pak vektorem T získáme údaje o citlivosti navrhovaného systému na změny v modulu i. Je-li $a_j=1$ pro všechna j, pak získáme přehled o celkové konektivitě návrhu systému. Pak výraz $m = (\sum(t_i - 1))/n$ dává míru složitosti návrhu systému. Čím je m vyšší, tímje udržovatelnost nižší. Je-li m nedefinované (řada v T nekonverguje) nebude program nikdy dokončen. Hodnoty T,m a t_i dávají možnost i pro odhad testovacích a udržovacích nákladů. Řada $1 + P + P^2 + ...$ konverguje, jsou-li vlastní hodnoty matice P v uzavřeném intervalu (0,1). Je to podmínka pro životaschopnost programu, protože jinak jedna změna způsobí nekonečný počet změn.

Matice propojení má při dodržení popisované struktury jednoduchý tvar (obr.3.)Má úroveň podprogramu,tzn že obsahuje řádky pro moduly i data každého podprogramu i třídy. Prvky lze vyjádřit pomocí informací dostupných na konci návrhu systému na vyšší úrovni a to pomocí této symboliky:

* (i,j) znamená podprogram j třídy i. i=E znamená řídicí třídu a j=0 znamená odkaz na třídu.(např.(i,0) je odkaz na třídu modulů nebo dat třídy i).

*$N_M(i,j)$ je počet modulů (i,j).(Např. $N_M(E,1)$ je počet modulů podprogramu 1 řídicí třídy)

$N_D(i,j)$ je počet datových objektů (i,j)
$R(i,j)$ je počet alokací datových objektů (i,j) do modulu (i,j). (Alokací se zde rozumí fakt, že modul se smí odkazovat na datové objekty)

Prvky propojovací matice se pak aproximují těmito vztahy:
$M(i,j) = m_i \, N_M(i,j)$, kde $m_i = \frac{P}{NPM}$, kde P je pravděpodobnost, že změna v jednom modulu dané úrovně způsobí změnu v jiném modulu téže úrovně, když úroveň obsahuje max. počet modulů, a NPM je max. počet modulů na této úrovni.

$D(i,j) = d_i \, N_D(i,j)$, kde $d_i = \frac{P}{NPDO}$, kde P je pravděpodobnost, že změna v jednom datovém objektu způsobí změnu v jiném datovém objektu na téže úrovni, když úroveň obsahuje max. počet datových objektů, a NPDO je max. počet datových objektů na této úrovni.

$X(i,j) = x_i(R(i,j) - N_D(i,j))/(N_M(i,j) \, N_D(i,j))$, kde $N_M(i,j) \neq 0$; v jiném případě se bere $X(i,j) = 0$.
x_i je pravděpodobnost, že změna v modulu i,j způsobí změnu v datovém objektu(i,j), když všechna data objektů (i,j) jsou alokována do všech modulů (i,j).

4. Příklad

Obr.4 ukazuje příklad propojovací matice, v níž jsou dvě problémové a jedna řídicí třída, každá se dvěma podprogramy. Typické hodnoty pro parametry jsou uvedeny v Tab.1. Konstanty m,d a x vyjadřují váhy udělené každému typu propojení a jsou navrženy tak, aby vytvářely pravděpodobnosti, jejichž suma je menší než 1. Tab.2. uvádí tři případy vektoru T pro tři varianty návrhu. První varianta je pro matici na obr.4, druhá pro případ, kdy se data řídicí třídy zredukovala přesunem 80% dat do problémové třídy a třetí, v níž se všechna říd.třídy a 2.třídy vynechala. Přesun dat ovlivnil $t_{10} - t_{18}$. Tvláště t_{18} se žádoucím způsobem zredukoval a m se snížilo o 25%. Ostatní prvky se změnily jen málo, protože zlepšení způsobené méně globálními daty převažuje nad degradací způsobenou větším počtem problémových dat. Je to jednoduchý projev staré pravdy, že data mají být pokud možno lokální. Ve třetí variantě je interakce vyloučených datových tříd nulové (viz t_{17} a t_{18}), menší redukce je i u jiných prvků a m se opět zmenšilo o 61%.

5. Systém STEP

V závěru publikace /1/ se autoři zmiňují o systému STEP ("Structured Techniques for Engineering Projects") což je soubor metod , pravidel a nástrojů využívaných v etapě vývoje programového vybavení. Systém STEP je strukturovaná kombinace lidské činnosti, nástrojů , datových bází a interaktivních počítačových prostředků, které podporují činnosti při řízení projektu, programování, kontrole vlastností výsledného produktu i při odhadech nákladů na vývoj.

6. Literatura

/1/ Rosene,A.F.,Connolly,J.E.,Bracy,K.M.: Software Maintainability, What It Means and How to Achieve It.
IEEE Transactions on reliability,Vol.R-30,No.3,August 1981

/2/ in /1/ Haney,F.M.: Module conection analysis - A tool for scheduling software debugging activities.
AFIPS Conf.Proc. Vol 41 1972

/3/ in /1/ Haney,F.M.: The architecture of software
Data Base Magazine, Vol5,No.1.1973

Pozn. S ohledem na úsporu místa jsou obrázky uvedeny v pořadí 2,1,3,4.

i,j,	M	D	X
E,0	0.025	0.04	0.0008
1,0	0.035	0.05	0.0006
2,0	0.05	0.075	0.0004
E,1	0.0025	0.015	0.0025
E,2	0.001	0.005	0.01
1,1	0.0025	0.002	0.0025
1,2	0.005	0.004	0.0011
2,1	0.0025	0.005	0.0025
2,2	0.0025	0.01	0.0025

Tab.1. Hodnoty parametrů

j	spojení podle obr.3	redukce dat řídicí třídy o 80%	vyloučení dat třídy
1	1.078917	1.078445	1.077214
2	1.080047	1.079628	1.078507
3	1.098834	1.098360	1.096576
4	1.098849	1.098374	1.096589
5	1.037765	1.037266	1.036402
6	1.044257	1.043334	1.042464
7	1.148973	1.148568	1.147459
8	1.203055	1.202629	1.200974
9	1.252247	1.251791	1.250671
10	1.124140	1.085421	1.004702
11	1.124882	1.086086	1.005207
12	1.166169	1.128689	1.007780
13	1.172059	1.134390	1.012870
14	1.078254	1.018895	1.017859
15	1.073325	1.016534	1.015502
16	1.233564	1.223619	1.001916
17	1.334332	1.325350	1.000000
18	1.437900	1.018230	1.000000
a	0.15475	0.11531	0.060705

Tab 2. Hodnoty vektoru T a ukazatele konektivity a pro tři případy

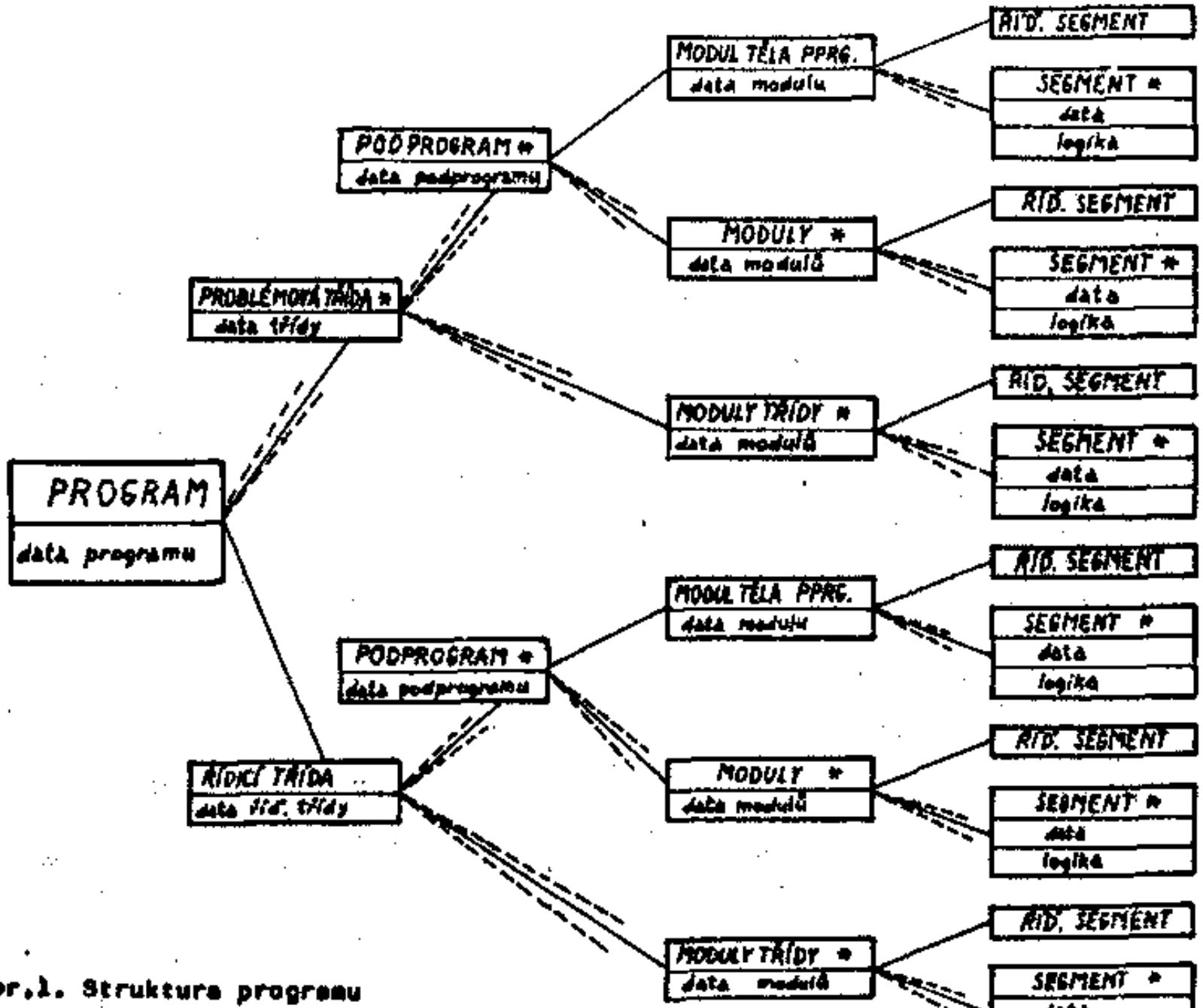
Obr.2. Tabulka volání

"C" ... Y může volat X

"A" ... Y může aktivovat X

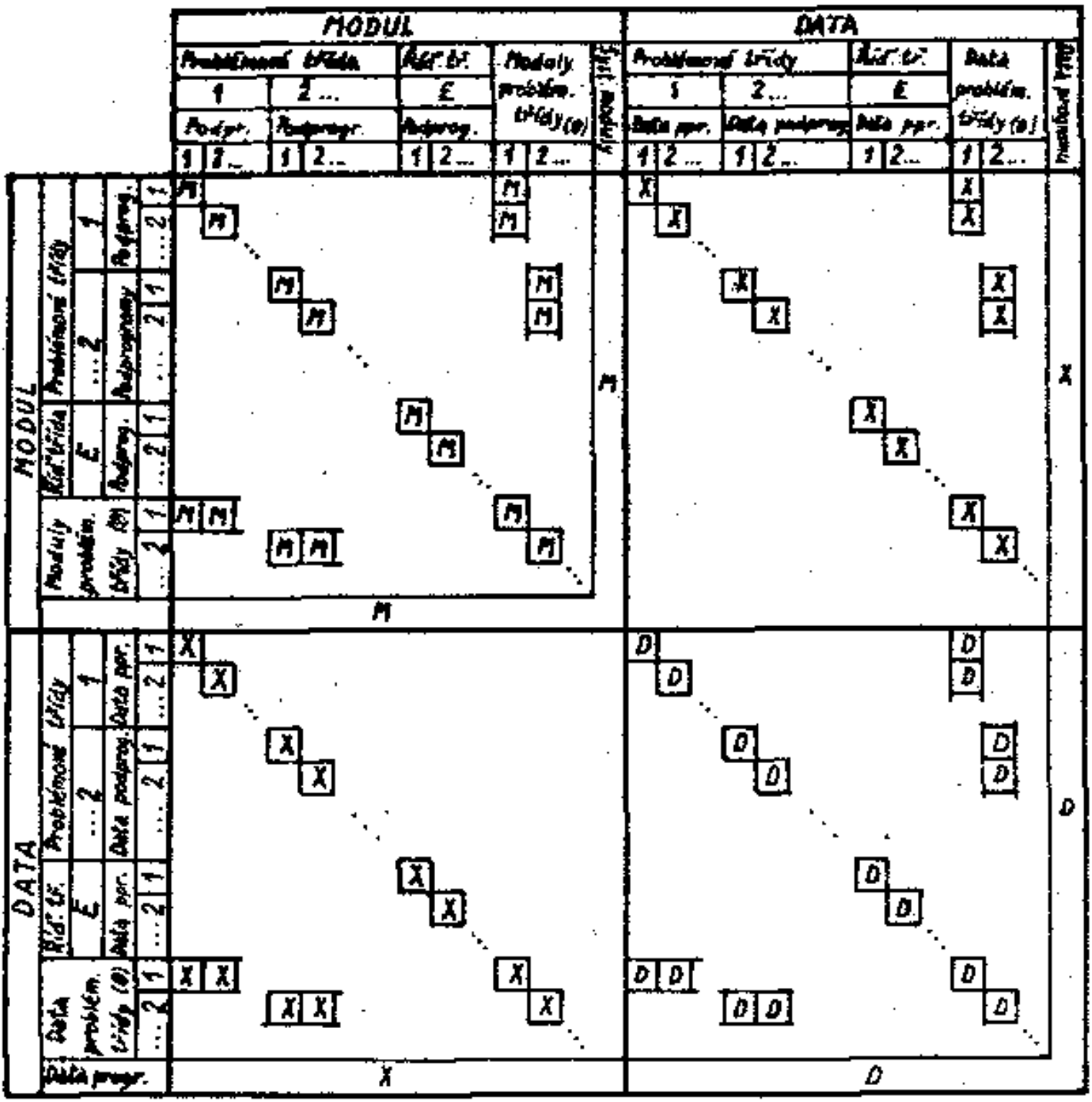
Prázdná kolonka znamená, že neexistují volací ani aktivační vztahy mezi Y a X

Y \ X			PROBLÉMOVÁ TŘÍDA			ŘÍDICÍ TŘÍDA		
			podprogram		modul třídy	podprogram		rychlý servisní modul
			modul těla	modul		modul těla	modul	
PROBLÉMOVÁ TŘ.	podprogram	modul těla		C	C			C
		modul		C	C			C
	modul třídy				C			C
ŘÍDICÍ TŘÍDA	podprogram	modul těla	A			A	C	C
		modul	A			A	C	C
	rychlý servisní modul							C



Obr.1. Struktura programu

Hvězdička za označením objektu znamená, že se objekt může vyskytnou opakovaně na téže úrovni (zdůrazněno čárkovanými spojkami)



Obr.3. Tvar matice propojení

$$P = \begin{array}{|c|c|} \hline M & X \\ \hline X_T & D \\ \hline \end{array}$$

$$M = \begin{array}{ccccccccc}
M\ 1,1 & 0 & 0 & 0 & 0 & 0 & M\ 1,0 & 0 & M\ E,0 \\
0 & M\ 1,2 & 0 & 0 & 0 & 0 & M\ 1,0 & 0 & M\ E,0 \\
0 & 0 & M\ 2,1 & 0 & 0 & 0 & 0 & M\ 2,0 & M\ E,0 \\
0 & 0 & 0 & M\ 2,2 & 0 & 0 & 0 & M\ 2,0 & M\ E,0 \\
0 & 0 & 0 & 0 & M\ E,1 & 0 & 0 & 0 & M\ E,0 \\
0 & 0 & 0 & 0 & 0 & M\ E,2 & 0 & 0 & M\ E,0 \\
M\ 1,0 & M\ 1,0 & 0 & 0 & 0 & 0 & M\ 1,0 & 0 & M\ E,0 \\
0 & 0 & M\ 2,0 & M\ 2,0 & 0 & 0 & 0 & M\ 2,0 & M\ E,0 \\
M\ E,0 & M\ E,0 & M\ E,0 & M\ E,0 & M\ E,0 & M\ E,0 & M\ E,0 & M\ E,0 & M\ E,0
\end{array}$$

$$D = \begin{array}{ccccccccc}
D\ 1,1 & 0 & 0 & 0 & 0 & 0 & D\ 1,0 & 0 & D\ E,0 \\
0 & D\ 1,2 & 0 & 0 & 0 & 0 & D\ 1,0 & 0 & D\ E,0 \\
0 & 0 & D\ 2,1 & 0 & 0 & 0 & 0 & D\ 2,0 & D\ E,0 \\
0 & 0 & 0 & D\ 2,2 & 0 & 0 & 0 & D\ 2,0 & D\ E,0 \\
0 & 0 & 0 & 0 & D\ E,1 & 0 & 0 & 0 & D\ E,0 \\
0 & 0 & 0 & 0 & 0 & D\ E,2 & 0 & 0 & D\ E,0 \\
D\ 1,0 & D\ 1,0 & 0 & 0 & 0 & 0 & D\ 1,0 & 0 & D\ E,0 \\
0 & 0 & D\ 2,0 & D\ 2,0 & 0 & 0 & 0 & D\ 2,0 & D\ E,0 \\
D\ E,0 & D\ E,0 & D\ E,0 & D\ E,0 & D\ E,0 & D\ E,0 & D\ E,0 & D\ E,0 & D\ E,0
\end{array}$$

$$X = \begin{array}{ccccccccc}
X\ 1,1 & 0 & 0 & 0 & 0 & 0 & X\ 1,0 & 0 & X\ E,0 \\
0 & X\ 1,2 & 0 & 0 & 0 & 0 & X\ 1,0 & 0 & X\ E,0 \\
0 & 0 & X\ 2,1 & 0 & 0 & 0 & 0 & X\ 2,0 & X\ E,0 \\
0 & 0 & 0 & X\ 2,2 & 0 & 0 & 0 & X\ 2,0 & X\ E,0 \\
0 & 0 & 0 & 0 & X\ E,1 & 0 & 0 & 0 & X\ E,0 \\
0 & 0 & 0 & 0 & 0 & X\ E,2 & 0 & 0 & X\ E,0 \\
0 & 0 & 0 & 0 & 0 & 0 & X\ 1,0 & 0 & X\ E,0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & X\ 2,0 & X\ E,0 \\
X\ E,0 & X\ E,0 & X\ E,0 & X\ E,0 & X\ E,0 & X\ E,0 & X\ E,0 & X\ E,0 & X\ E,0
\end{array}$$

Obr.4. Příklad matice P