Josef Černý
Ústav výpočtové techniky ČVUT Praha

PRINCIPY NÁVRHU RT SYSTÉMŮ

Příspěvek se zabývá některými základními etapami návrhu systémů pro zpracování v reálném čase. Alternativní řešení jsou diskutována především vzhledem k požadavku dodržení stanovené doby odezvy jednotlivých řídících akcí. Je navržena metoda vytváření RT systémů na základě funkční a časové klasifikace procesů.

1. ÚVOD

Prudký rozvoj počítačových systémů pro zpracování v reálném čase (zkráceně real-time, RT), k němuž v současné době dochází, vyžaduje kvalitativní změnu v dosavadním (v našich podmínkách stále ještě spíše amatérském) přístupu k této problematice. Tento příspěvek uvádí některé základní principy návrhu RT systémů, které vycházejí z praktických i teoretických zkušeností získaných v ÚVT ČVUT při realizaci několika RT systémů. Základní seznámení s problematikou, jejíž znalost je předpokládána, nalezne čtenář v (3), kde je též uveden poměrně vyčerpávající seznam další literatury.

I když je velmi obtížné pojem real-time přesněji vymezit, je možné říci, že RT systém řídí okolí přijímáním dat, jejich

zpracováním a prováděním akcí nebo vracením výsledků přiměřeně rychle, za účelem ovlivňování činnosti okolí v tomto čase. Základním stavebním prvkem RT systémů je proces, tj. sekvenční program, jehož operace však mohou vůči operacím ostatních procesů probíhat souběžně. Specifikací interface mezi jednotlivými procesy v samostatném programovém celku dostáváme základní strukturu RT systému, strukturu "jádro-procesy".

Návrh a realizace RT systému probíhá obvykle v několika etapách. Nejprve je na základě obecných vlastností předpokládané aplikační oblasti navržen charakter systému a zásady interface mezi (řídícím) systémem a vnějším (řízeným) prostředím, tj. způsob organizace a synchronizací řídících (aplikačních) procesů. Základní funkční charakteristiky systému jsou postupně zpřesňovány a doplňovány tak, jak postupují práce na návrhu jak uživatelského jazyka, tak jádra systému, tj. základního programového vybavení umožňujícího existenci, synchronizaci a kooperaci jednotlivých procesů.

Po implementaci a odladění jádra (včetně jeho vazeb na operační systém hostitelského počítače) je tedy k dispozici RT operační systém (virtuální počítač) schopný užití v dané aplikační oblasti. Vytvoření RT systému pro konkrétní aplikaci vychází z detailní systémové analýzy řešeného fyzického systému, která je podkladem pro vypracování jednotlivých procesů a jejich implementaci pod RT operačním systémem.

Příspěvek se zabývá těmi etapami návrhu, které podle našeho názoru podstatně ovlivňují výsledný charakter systému, především jeho schopnost reagovat včas (tj."přiměřeně rychle") na podněty vnějšího prostředí (tzn.respektovat požadovanou dobu odezvy jednotlivých procesů).

2. METODY ORGANIZACE A SYNCHRONIZACE RT PROCESŮ

2.1. ORGANIZACE

Podstatou řízení je zachycení činností řízené soustavy ve stavových změnách řídícího systému, tj. jednotlivých procesů

Korespondence mezi fyzickými ději řízené soustavy a odpovídajícími RT procesy a organizace těchto procesů může být řešena v zásadě dvěma způsoby.

První z nich spočívá v tom, že z vnějších procesů řízené soustavy vytvoříme konečný (a relativně malý) počet skupin, v nichž vnější procesy funkčně souvisejí a časově se nepřekrývají (např. postupné opracování výrobků na jednom zařízení). Pak můžeme každé takovéto skupině přiřadit při vytváření řídícího systému jediný vnitřní proces, který existuje po celou dobu činnosti systému, jenž je tedy z hlediska počtu existujících RT procesů statický.

Má-li řízená soustava charakter systému hromadné obsluhy (tzn. může-li v ní probíhat současně několik shodných činností), existují pak v statickém řídícím systému byť formálně různé, ale fakticky stejné řídící procesy, což se může právem zdát zbytečné.

V tom případě seskupíme vnější procesy pouze z hlediska jejich funkční podobnosti bez ohledu na časovou posloupnost jejich skutečného vykonávání a každé takovéto skupině přiřadíme nikoliv samostatný proces, nýbrž jen řídící algoritmus. Počet současně existujících vnitřních procesů se společným řídícím algoritmem (tzv. exemplářů) se pak v systému dynamicky mění podle počtu současně probíhajících vnějších činností patřících do jedné skupiny. Typickým případem je popis operací na několika současně pracujících zařízeních jedním řídícím algoritmem. Případné funkční odchylky jednotlivých fyzických činností můžeme řešit užitím lokálních parametrů při vytváření jednotlivých exemplářů.

Statické systémy (s pevným počtem RT procesů) mají oproti dynamickým výhodu v jednodušší realizaci funkcí řídících pseudoparalelní chod a umožňují použití strukturovanějších jazykových prostředků s možností kontrol v době překladu. Dynamické systémy (s proměnným počtem procesů) mají naproti tomu užší vztah k realitě řízené soustavy, snáze se modifikují a jejich použití je univerzálnější.

2.2. SYNCHRONIZACE

Synchronizací rozumíme obecný termín pro jakékoliv omezení pořadí provádění jednotlivých operací nebo jejich částí. Toto omezení je definováno synchronizační podmínkou, která popisuje situaci, za níž je možné daný proces (nebo jeho část) provést, resp. mu v pseudoparalelních systémech přidělit virtuální procesor.

Každý vytvořený proces (jeho virtuální procesor) může být přerušován (inaktivován) a znovu aktivován, ať už explicitně (programově) nebo implicitně uvnitř jádra. To odpovídá dvěma základním stavům procesu : aktivní, inaktivní.

Obecným synchronizačním prostředkem užívaným v RT systémech je aktivační podmínka, přesněji řečeno možnost podmíněné aktivace jistého procesu. Aktivační podmínka umožňuje zajistit:

1) synchronizaci činnosti řídících procesů s ději probíhajícími v řízené soustavě,
2) vzájemnou synchronizaci spolupracujících a soutěžících RT procesů.

V okamžiku splnění aktivační podmínky (tj. výskytu jisté synchronizační události, které může nastat implicitně z kterékoliv části systému, je odpovídající čekající proces připraven k aktivaci. Je zřejmé, že v daném okamžiku může být takovýchto procesů více. Protože je však většinou můžeme (vzhledem k omezenému počtu procesorů) aktivovat pouze postupně, je nutné provádět rozvrhování čekajících procesů.

Synchronizační primitivy bývají naprogramovány v jádře systému a při jejich explicitním či implicitním vyvolání se provádí odejmutí či přidělení procesoru, záznam o stavech procesu a vlastní rozvržení procesů.

Formálně si můžeme aktivační podmínku představit jako datovou strukturu se dvěma operacemi ZÁPIS a TEST. Operace ZÁPIS slouží ke změně hodnoty dat aktivační podmínky (výsledkem činnosti některého vnějšího nebo vnitřního procesu), operace TEST

zjišťuje stav podmínky (tj. stav odpovídajícího čekajícího procesu). Pozitivní výsledek provedení operace TEST znamená přechod čekajícího procesu mezi připravené, tj. přidělení virtuálního procesoru. Operace TEST může být provedena :
1) z rozhodnutí jádra systému (centralizovaně),
2) jako okamžitý důsledek operace ZÁPIS (distribuovaně).

Volba způsobu testování hraje v návrhu RT systému důležitou roli, neboť souvisí s volbou základních datových struktur i metody rozvrhování RT systému.

Centralizované testování logicky odděluje operace TEST a ZÁPIS, tj. změnu hodnoty podmínky od zjištění, že k této změně došlo. Tento "odklad testování" zřejmě zvyšuje dobu odezvy čekajícího procesu, čímž klesá jeho schopnost co nejrychleji reagovat na synchronizační událost.

Distribuované testování naproti tomu umožňuje čekajícímu procesu přejít mezi připravené ihned v okamžiku splnění jeho aktivační podmínky. Na druhé straně se však (u složitých, často se měnících podmínek) zhoršuje využití procesoru zbytečným neúspěšným testováním (důsledek operace ZÁPIS, která nevede ke splnění podmínky).

Užití distribuovaného testování má tedy své opodstatnění jen v případě jednoduchých aktivačních podmínek, kdy umožňuje (spolu s vhodným rozvrhováním) dosáhnout velmi malé skutečné doby odezvy. V ostatních případech však dáme přednost testování centralizovanému.

3. ROZVRHOVÁNÍ RT PROCESŮ

Protože ve většině případů není možné zabezpečit okamžité přidělení volného procesoru každému spustitelnému procesu, je nutné provádět jejich aktivaci postupně, tzn. rozvrhovat je. Pro rozvrhování spustitelných procesů je třeba zvolit strukturu množiny připravených procesů (čekajících na uvolnění procesoru). Ověřeným způsobem spořádaného čekání je FRONTA, řešená případně

na několika prioritních úrovních. Vlastní rozvrhování je obecně poměrně složitým samostatným problémem, je možné použít nejrůznějších rozvrhovacích strategií. My se však zaměříme na otázku přerušitelnosti nebo nepřerušitelnosti aktivního procesu (tj. na preemptivní či nepreemptivní techniku rozvrhování) ve vztahu k velikosti skutečné doby odezvy jednotlivých RT procesů systému.

Nepreemptivní rozvrhování vychází z předpokladu, že procesor zůstává aktivnímu procesu přidělen tak dlouho, dokud není tímto procesem uvolněn (jádro systému nemá možnost aktivnímu procesu procesor odebrat a přidělit jej v případě potřeby jinému). V takovém systému je tedy možné zahájit vždy jen jediný proces a spustitelný (i když prioritnější) proces je nucen čekat na dokončení právě aktivního procesu, což může zvýšit jeho skutečnou dobu odezvy.

Metodu rozvrhování lze však navrhnout i tak, aby mohlo procesor sdílet několik již zahájených (rozpracovaných) procesů. To ovšem znamená, že jádro systému musí mít možnost měnit přidělení procesoru v průběhu zahájeného procesu, tzn. přidělit procesor vždy takovému procesu, který má (vzhledem ke zvolené rozvrhovací strategii) nejvyšší prioritu. Příchod prioritnějšího požadavku (spustitelného procesu) tedy znamená okamžitou změnu přidělení procesoru-přerušení činnosti dosud aktivního procesu. Preempce umožňuje snižovat doby odezvy prioritních procesů, ovšem za cenu zvyšování doby odezvy procesů s nižší prioritou, které jsou poměrně často přerušovány. Navíc vyžaduje složitější metody organizace dat, které zajišťují zachování průběhu operací procesu za bodem jeho přerušení.

4. NÁVRH JÁDRA RT SYSTÉMU

Řekli jsme, že jádro RT systému obsahuje základní programové vybavení umožňující existenci, synchronizaci a kooperaci (vnitřních) procesů, tzn. specifikaci interface mezi nimi, a zároveň (ve formě procedur) realizuje všechny RT konstrukce na-

vržemého uživatelského jazyka. Pro účely našeho článku jsou podstatné především následující funkce jádra :

1) aktivační mechanismus, který v okamžiku uvolnění procesoru vybere z množiny připravených procesů nový požadavek na spuštění a přidělí mu volný procesor,
2) připravovací mechanismus, který aktualizuje množinu připravených procesů zařazováním čekajících procesů se splněnou aktivační podmínkou (provedením operace TEST),
3) mechanismus aktualizace dat RT systému, který na základě výskytu události aktualizuje data jednotlivých aktivačních podmínek (provedením operace ZÁPIS),
4) způsob organizace paměti (především zajištění konzistence lokálních dat a výlučného přístupu ke globálním sdíleným datům).

Základní typ jádra RT systému je určen v prvé řadě způsobem realizace pseudoparalelního chodu procesů, tj. metodou rozvrhování. Budeme proto hovořit o jádru preemptivního (resp. nepreemtivního) typu. Proces vytváření jádra pak spočívá ve výběru mechanismů, datových struktur a jejich vzájemných vazeb, které charakter zvoleného typu jádra podpoří, a v realizaci RT konstruktů uživatelského jazyka (o nichž se zmiňovat nebudeme).

4.1. JÁDRO PREEMPTIVNÍHO TYPU

Preemptivní rozvrhování umožňuje změnu přidělení procesoru v libovolném časovém okamžiku novým rozvržením procesů (důsledek synchronizační události). Mechanismy jádra musí být proto navrženy tak, aby se výskyt události znamenající splnění jisté aktivační podmínky projevil okamžitě v přechodu odpovídajícího čekajícího procesu mezi připravené a umožnil tak (po novém rozvržení) jeho případnou aktivaci. To znamená, že výskyt (vnější) události způsobí přerušení aktivního procesu, zpracování události (aktualizace dat příslušné aktivační podmínky a její následné distribuované testování) a nové rozvržení systému (s ak-

tualizovanou množinou připravených procesů).

Jádro systému preemptivního typu tedy umožňuje bezprostřední reakci RT systému na události vyžadující enormě nízkou dobu odezvy. Časté přerušování aktivního procesu (v případě vyšší četnosti výskytu události) však vede ke zvětšování overheadu jádra, což zpětně nepříznivě ovlivňuje skutečnou doby odezvy jednotlivých procesů. Preempce navíc vyžaduje takovou organizaci paměti, která zabezpečí úschovu lokálních dat procesu v okamžiku jeho přerušení a vzájemné vyloučení přístupu k jeho sdíleným datům. To vede (v důsledku použití známých synchronizačních metod jako jsou např. monitory, kritické sekce, semafory apod.) k požadavku prevence systému před možným zablokováním, případně k potřebě nalezení vhodné struktury sdílených dat, což může být obecně velmi obtížné. Ke zvyšování overheadu přispívají i často výměny programů (dat) mezi vnější a operační pamětí (swapping) v okamžiku jejich aktivace.

4.2. JÁDRO NEPREEMPTIVNÍHO TYPU

Nepreemptivní rozvrhování dovoluje rozpracovat vždy jen jediný proces, jemuž je procesor výhradně přidělen až do okamžiku, kdy tento aktivní proces skončí nebo se z vlastní vůle pozastaví (programová suspendace). Tím dojde k uvolnění procesoru a novému rozvržení připravených procesů. Protože činnosti aktivního procesu nemůže být přerušena příchodem nového požadavku na spuštění, stačí aktualizovat množinu připravených procesů až v okamžiku nového rozvržení. To se samozřejmě musí projevit i ve způsobu zpracování událostí. Výskyt každé události je sice zaregistrován, avšak její zpracování (operace ZÁPIS) je odloženo. V okamžiku, kdy má dojít k aktualizaci množiny připravených procesů, jsou všechny doposud zaregistrované události zpracovány a je proveden výběr procesů se splněnou aktivační podmínkou. Užití centralizovaného testování tak na jedné straně odstraňuje zbytečná zjišťování stavu jednotlivých aktivačních podmínek, nepřerušitelnost aktivního procesu však na straně druhé snižuje schop-

nost jádra zajistit včasné provedení procesů s malou požadovanou dobou odezvy, což je také jedinou podstatnou nevýhodou jádra tohoto typu. Jednoduché mechanismy řízení a synchronizace, organizace paměti nevyžadující ochranu sdílených dat, menší overhead jádra a v neposlední řadě i jednodušší struktura a snažší odladitelnost činí (ve většině aplikací) použití jádra nepreemptivního typu výhodnějším s vyjímkou těch situací, které vyžadují enormě rychlé reakce řídícího systému. Tam se bez použití preempce patrně neobejdeme. Možné řešení této situace ukážeme v následující kapitole.

5. NÁVRH APLIKAČNÍCH PROCESŮ

Ve složitých systémech (jako jsou například biologické aktivity) můžeme pozorovat množství složitě strukturovaných činností. Podstatou velké účinnosti těchto systémů je stupňovitost organizace jejich řídících funkcí. Řízení je sice v zásadě centralizované (mozek), avšak jednotlivé dílčí procesy jsou podle své důležitosti hierarchicky uspořádány. Zároveň se však podílejí na řízení určitého funkčního celku a jistým způsobem spolu souvisejí. Každou řídící činnost tedy můžeme charakterizovat jak jejím postavením v hierarchické řídící struktuře, tak příslušností do některé funkční části systému. Z těchto do jisté míry poměrně protichůdných hledisek budeme vycházet i při návrhu struktury RT systému a jednotlivých procesů.

5.1. FUNKČNÍ ZÁVISLOST

Činnost každé soustavy je souhrnem dějů v ní probíhajících a navzájem více či méně funkčně závislých. Úkolem systémové analýzy je tyto děje najít, stanovit jejich vzájemné vztahy a popsat je odpovídajícími řídícími procesy. Obvykle přitom postupujeme tak, že řízenou soustavu členíme postupně metodou shora dolů na jednotlivé celky obsahující ty děje, které spolu na dané úrovni analýzy souvisejí. Opakovaným členěním těchto celků můžeme dospět až na

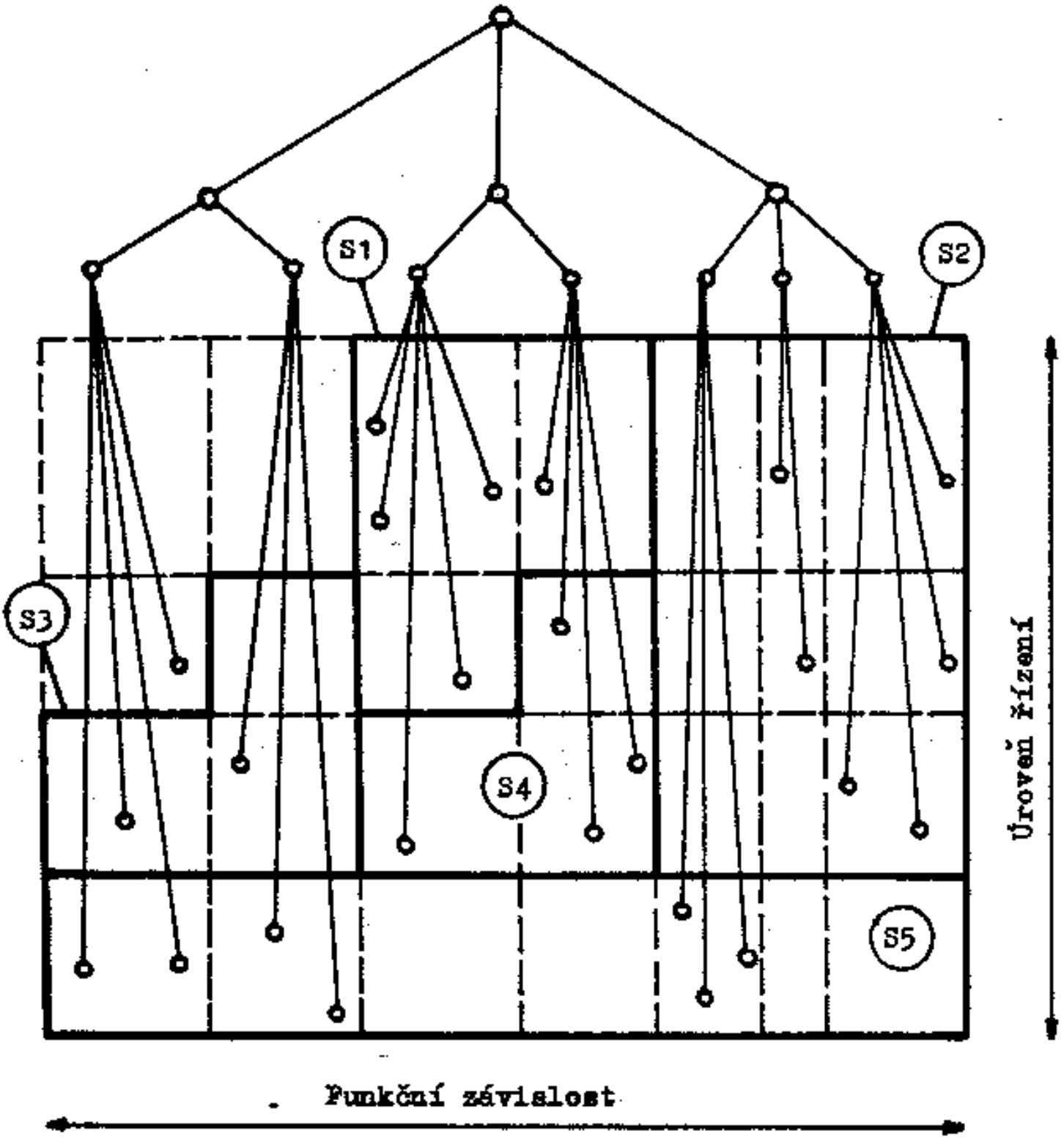
takovou úroveň, na níž popíšeme jednotlivé činnosti příslušnými řídícími procesy. Celý postup je možné znázornit stromem, jehož listy jsou jednotlivé řídící procesy a kořeny podstromů definují logické celky (subsystémy) získané na předchozích úrovních analýzy. Čím je vzdálenost podstromu od kořene stromu větší, tím byl odpovídající celek vytvořen na detailnější úrovni systémové analýzy a obsahuje proto funkčně závislejší procesy. Nazveme-li tuto vzdálenost hladinou podstromu, můžeme říci, že míra funkční závislosti procesů je určena hladinou nejmenšího stromu obsahujícího tyto procesy.

Funkční závislost procesů se projevuje zejména v množství vzájemně vyměňovaných informacích (dat). Tato výměna může probíhat v zásadě dvěma odlišnými způsoby : buď procesy pracují nad společnými daty, nebo komunikují. Tomu odpovídá i strukturalizace paměti z hlediska procesů. Buď mají procesy společnou paměť nebo jen vlastní soukromá data. Je zřejmé, že komunikace zvyšuje samostatnost procesů a podporuje modulární přístup při vytváření systému. Na druhé straně jsou však tyto zřejmé přednosti oslabovány duplikováním přenášené informace a větší složitostí podpůrných prostředků systému. Je tedy přirozené vytvářet společná data jen pro takové procesy, jejichž výměna informací (a tedy i funkční závislost) je dostatečně velká, a ostatní případy řešit komunikací.

Rozdělení systému do jednotlivých funkčních celků tedy provedeme tak, aby procesy se společnými daty patřily vždy do téže funkční části. Takovéto zvýšení samostatnosti procesů vede k posílení modularity celého systému.

5.2. ÚROVEŇ ŘÍZENÍ

Mechanismy, jimiž je živý organismus bezprostředně svázán s vnějším prostředím, jsou většinou nejnižšími články hierarchické řídící struktury a vyznačují se nízkou dobou odezvy. Jejich úkolem je okamžitě reagovat na stav okolního prostředí a o svém zákroku podat informaci vyššímu článku řízení. Se zvyšující se



Na obr.: Výsledná modulární struktura RT systému
(subsystémy S1 - S5)

úrovní řízení roste množství zpracovávaných informací sloužících jako podklady pro složitější rozhodování, současně však roste velikost požadované doby odezvy. Je přirozené požadovat, aby i činnost RT systémů zachovávala podobný princip, tzn. aby procesy se složitější strukturou nevyžadovaly enormě rychlou obsluhu. Zkušenosti ukazují, že tomuto požadavku lze většinou bez velkých obtíží vyhovět. S růstem inteligence procesu (jeho logické složitosti) a množství jím zpracovávaných dat se zvyšuje jeho řídící úroveň a tím i požadovaná doba odezvy.

Rozdělme nyní jednotlivé procesy systému do navzájem disjunktních tříd tak, aby každá třída obsahovala jen procesy se srovnatelnou dobou odezvy, tedy procesy na přibližně stejné řídící úrovni. Přiřadíme-li každé třídě (subsystému) určitou prioritu, rozpadne se rozvrhování celého systému na dvě úrovně:
1) rozvrhování jednotlivých procesů uvnitř třídy,
2) rozvrhování jednotlivých tříd v rámci celého systému.

Každý subsystém obsahuje vlastní jádro zajišťující rozvrhování a interface mezi procesy subsystému, zároveň však existuje centrální jádro, které realizuje rozvrhování jednotlivých subsystémů a interface mezi nimi.

Ukázali jsme, že na volbu jednotlivých způsobů rozvrhování má rozhodující vliv velikost požadované doby odezvy. V každém subsystému se tedy rozhodneme pro takový typ jádra, který je z hlediska procesů subsystému nejvýhodnější. Preemptivní jádro volíme většinou jen u subsystémů s nejvyšší prioritou (případně jen u jednoho takového). Tím umožníme obsluhu kritických požadavků a zároveň zachováme výhody nepreemptivního rozvrhování v ostatních subsystémech. Rozvrhování celého systému navrhneme tak, aby prioritnější subsystém mohl v případě potřeby přerušit činnost jiného subsystému s nižší prioritou (tj. preemptivně).

5.3. NÁVRH STRUKTURY RT SYSTÉMU

Ukázali jsme, že klasifikace procesů na základě velikosti požadované doby odezvy je vhodným prostředkem pro stanovení způsobu plánování respektujícího požadované doby odezvy RT procesů. Analýza funkční závislosti vede naopak k vytvoření poměrně samostatných částí systému, ovšem bez ohledu na požadovanou dobu odezvy. Naším cílem je vytvoření takové struktury systému, která by byla optimálním kompromisem mezi těmito dvěma hledisky. Třídy vytvořené z procesů se srovnatelnou dobou odezvy proto modifikujeme na základě jejich funkční závislosti tak, aby pokud možno obsahovaly jednotlivé funkční celky systému (viz. obr.). Povaha výsledné struktury je dána tím, které hledisko hrálo při jejím vytváření podstatnější roli. Jsou-li požadované doby odezvy funkčně závislých procesů srovnatelné, odpovídá výsledná struktura do značné míry funkčnímu členění systému, v opačném případě má výrazné prioritní rysy. Vytvořené subsystémy jsou tedy do značné míry funkčně nezávislé a nevyžadují proto existenci společných dat; jejich interface tak můžeme řešit komunikací. Zároveň je však můžeme prioritně uspořádat a rozhodnout o nejvhodnějším způsobu rozvrhování.

6. ZÁVĚR

Návrh RT systému je hledáním takových struktur a prostředků jejich realizace, které by umožnily optimální kompromis mezi obecnými kritérii (jednoduchost, univerzálnost a efektivnost) tak, aby navrhovaný systém přirozeným způsobem pokryl předpokládanou aplikační oblast. Míra obecných kritérií se však v duchu "zákona zachování složitosti" přelévá a nelze proto navrhnout nejlepší systém ve smyslu absolutním, ale pouze systém optimální z hlediska vymezených aplikačních potřeb.

V článku je navržena základní struktura RT systému a popsán způsob jejího vytváření na základě velikosti požadované do-

by odezvy jednotlivých řídících procesů a jejich funkční závislosti. Zdá se, že tato metoda je nezávislá na aplikační oblasti, pro kterou má systém sloužit.

Míra univerzálnosti, efektivnosti a jednoduchosti výsledného systému je dána především vhodným seskupením procesů do jednotlivých subsystémů, volbou preemptivního či nepreemptivního rozvrhování a v neposlední řadě i konkrétní fyzickou realizací těchto subsystémů. Realizujeme-li například poměrně jednoduché, ale vzhledem k požadované době odezvy kritické řídící akce formou mikroprocesorů, můžeme navrhovaný systém podstatně zjednodušit a zvýšit jeho efektivnost, aniž by se tato změna projevila podstatněji v ostatních subsystémech.

Jednotný interface mezi jak RT systémem, jeho okolím, tak jednotlivými subsystémy (komunikace zasíláním zpráv posiluje modulární charakter systému, což usnadňuje jeho návrh, ladění i simulaci.

Protože nám omezený rozsah příspěvku nedovolil uvést ukázku aplikace uvedené metody v praxi, soustředíme na ni pozornost v našem vystoupení.

LITERATURA

(1) Brinch Hansen, P.: Operating system principles. Englewood Cliffs, Prentice-Hall, 1973

(2) Černý,J. - Dvořák,P. : Některé problémy návrhu RT systému, Sborník SOFSEM'80

(3) Demner,J. - Pučík,J.: Software pro zpracování v reálném čase, Sborník SOFSEM'79

(4) Král,J.- Demner,J. : New synchronization primitives for real-time programming. Acta Polytechnica, 2, 1978-1

(5) Kronental,M.: Towards the standardization of real-time operating system kernels. IRIA/ENI, SOCOCO'79, 1979

(6) Škarda,J.: Počítače pracující v reálném čase. SNTL,Praha,1978