OBECNÉ ZÁSADY TVORBY SPOLEHLIVÝCH ASŘTP V REÁLNÉM ČASE

Zdeněk Vaculín, Jaroslav Šolek, Ladislav Macháček
Kancelářské stroje, k. ú. o. OŘS Vsetín

Cílem příspěvku je nastínit několik základních pohledů na problém tvorby ASŘTP v reálném čase, seznámit s výsledky několikaletého úsilí pracovníků našeho střediska o vytvoření souboru univerzálních programů, které jsou součástí řídících systémů realizovaných ve strojírenství.

Požadavek univerzálnosti úzce souvisí s požadavkem spolehlivosti. Je obecně znám problém testování programových děl. I u "dobře" otestovaného programu po čase zjistíme, že všechny chyby nebyly odstraněny a je nutný zásah do zdrojového textu. Tím se mohou zanést další chyby, další opravy. Spolehlivost roste s úrovní testování. Univerzální programy, které jsou použity ve více aplikacích, jsou testovány v běhu systému, takže lze právem očekávat, že se jejich spolehlivost bude neustále zvětšovat.

Hlavním požadavkem při tvorbě ASŘTP je tedy spolehlivost. Zvětšení spolehlivosti docílíme opakovaným používáním programů. Základní úkol byl provést obecnou analýzu a dekompozici řídících systémů technologických procesů ve strojírenství, za účelem definování společných programů, použitelných ve všech aplikacích. Došlo k dekompozici celého systému na jednotlivé skupiny programů:
- programy zajišťující styk systému řízení s okolím
- programy realizující vlastní proces řízení
- programy pracující s daty.

Programy zajišťující styk s okolím jsou díky omezenému výběru hardwarových prostředků nejuniverzálnější. V zásadě jde o zajištění komunikace se sítí terminálů nebo dálnopisů a sběr binárních signálů z významného okolí řízeného objektu.

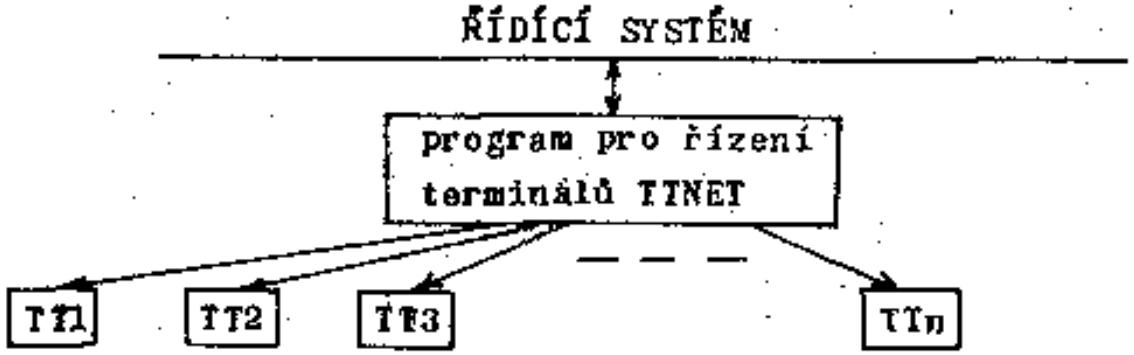
Programy realizující vlastní proces řízení již takto univerzální nejsou. Je to dáno úrovní automatizovatelnosti dané aplikace, různým obsahem a formátem dat, která jsou k dispozici pro daný řídící systém.

Program pracující s daty na disku, přestože formát dat je bohužel u všech aplikací různý, má shodné základní rysy. Osvědčilo se, aby s daty na disku pracoval jediný program. Ušetří se tak zbytečné komplikace se synchronizací přístupů k souborům. Požadavky na program systém frontuje a provádějí se tedy seriově. Při poruchách lze lépe zabezpečit integritu dat.

Díky stálému zvětšování operační paměti počítačů bude možné všechna data charakterizující řízený proces udržovat přímo v paměti počítače. Spolehlivost systémů se tím řádově zvětší, poněvadž poruchovost procesoru a paměti je proti diskům podstatně menší.

Jedním z nejuniverzálnějších programů je program zajišťující komunikaci se sítí terminálů. Cílem bylo sestrojit jeden program, který by byl schopen komunikace s proměnným počtem terminálů; tato možnost se určuje ve fázi překladu. Původní koncepce byla co terminál, to program. Jediný program je výhodnější, poněvadž při instalaci zabírá menší oblast dynamické paměti, není nutné složité řízení synchronizace. Požadavky na program jsou frontovány, takže platí "kdo dřív přijde, ten dřív mele".

Podíváme-li se na program z hlediska transakčního zpracování dat, zjistíme, že se vlastně jedná o jakýsi transakční monitor. Požadavky různých terminálů na provedení transakcí vznikají zcela nahodile a není mezi nimi přímá souvislost. Tyto požadavky program frontuje a tím zajišťuje integritu měněných dat, poněvadž většina transakcí je destruktivních, to znamená mění společná data.



Obr. 1: Program TTNET.

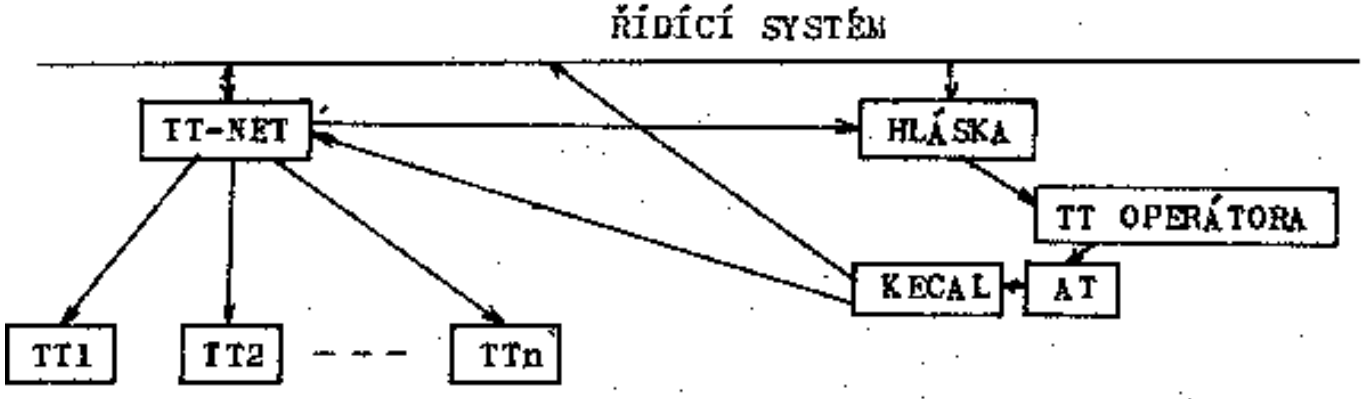
Z terminálů T1 až Tn přicházejí žádosti uživatelů o transakce. Uživatel volí kód, čili typ transakce. Poté interaktivním způsobem komunikuje. Komunikace se většinou skládá z několika otázek a odpovědí, je možné rozšíření programu a vyplňování formulářů. Program sám kontroluje syntaktickou správnost odpovědí. Po ukončení syntakticky správné komunikace, program posílá vykonverzovaná data řídícímu systému, který ověří jejich sémantickou správnost, a odešle zprávu na program buď potvrzující správnost komunikace, nebo informaci o chybě a kde se stala. Zprávy na program od řídícího systému se frontují pro každý terminál do zvláštní fronty. Toto frontování podle kódu (čísla terminálu) zprávy zabezpečuje rozšíření operačního systému.

Celý mechanismus běhu úlohy je založen na principu asynchronních přerušení při zařazení zprávy do fronty úlohy, při vstupu nevyžádaného znaku z terminálu (začátku dané komunikace), při dokončení vstupně výstupního požadavku (dokončení jednotlivých odpovědí) a při vyčerpání časového intervalu pro odpověď na vypsaný dotaz či zprávu na terminál. Program má také za úkol vypisovat různé zprávy na určený terminál. To, že uživatel bere vypsanou zprávu na vědomí, musí oznámit zmáčknutím kterékoliv klávesy. Toto opatření je nutné, poněvadž systém není schopen zjistit, zda daná zpráva byla na terminál vůbec vypsána a mohl by stále vypisovat jako na nulové zařízení.

Způsob využití asynchronních přerušení dovoluje nezávisle na délce jednotlivých odpovědí uživatelů paralelně komunikovat na n terminálech, aniž by se uživatelé navzájem omezovali.

Poněvadž bylo cílem vytvořit univerzální a spolehlivý program, bylo nutné omezit úpravy zdrojového textu na nejmenší míru. Pro různé aplikace je pouze potřeba zadefinovat ve fázi překladu počet terminálů a náplň jednotlivých transakcí pro určené terminály. Znamená to poze změnit tabulky konverzací - tedy změnit data.

V praxi vznikají situace, kdy dochází k poruše jednotlivých terminálů. Program má možnost přesměrovat požadavky na jiný, operátorem mu zadaný terminál.



Obr.2: Systém komunikace

Celý systém komunikace se kládá z popsaného již programu TTNET, programů HLASKA a KECAL. HLASKA zajišťuje výpis všech zpráv určených hlavnímu operátorovi a archivaci veškeré komunikace v terminálové síti. Parametry systému, tedy jeho chování, ovlivňuje hlavní operátor prostřednictvím příkazových souborů .AT a programu KECAL.

K systému komunikace nutno dodat, že program TTNET se připojí k daným terminálům tak, že není možné ovlivňování vlastností systému. Je povolena pouze množina těch transakcí, které jsou předem definovány v tabulkách programu pro daný terminál. Tím se po skušenostech našich pracovníků značně zvětší spolehlivost celého systému.

Vytvoření množiny univerzálních programů vytváří předpoklad pro pohodlnější vývoj systémů řízení metodou shora - dolů, není nutné se zabývat problémy komunikace řídícího systému s okolím a je možné se plně soustředit na návrh samotného systému řízení.

Další významnou zásadou je archivace veškeré komunikace. Všechny zprávy, které vstupují do a vystupují ze systému je nutno zaznamenat. Protože je mnohdy nemožné okamžitě vypisovat velké množství zpráv, je výhodné zprávy postupně archivovat na disk a v případě potřeby je vypisovat na tiskárnu. Všechny zprávy je potřeba označit aktuálním časem jejich vzniku, čímž se umožní zpětně určit jejich vzájemné souvislosti. Jen tak je totiž možno v průběhu zkušebního provozu systému analyzovat chyby jak systému tak obsluhy. Tím se ušetří množství konfliktů se zákazníkem o příčině vznikajících chyb a u systémových chyb se snáze odhalí

jejich příčina. Jako příklad lze uvést program HLÁSKA, který má dvě funkce. První je výpis zpráv na terminál dispečera, druhou je archivace zpráv jak programů KECAL a HLÁSKA, tak programu TISEI.

Další zásadou, kterou se snažíme dodržovat, je strukturované programování. Používáme soubor maker CODE vytvořený v našem středisku. Používání těchto maker umožňuje efektivní programování na úrovni asembleru, zároveň však zavedení struktur zpřehledňuje program. Není zanedbatelné také to, že u některých programátorů zavedení struktur kladně ovlivňuje jejich návrh programů, protože struktury samy vedou ke správné analýze řešeného problému.

Ze zásad strukturovaného programování vychází i návrh datových struktur. Je potřeba v předstihu navrhnout datové struktury tak, aby se v průběhu výstavby měnily co nejméně. Toto souvisí s návrhem celé koncepce řešení systému řízení a většinou spočívá na bedrech jediného pracovníka - tvůrce systému. Jeví se nám výhodné použít k definování datových struktur systémových maker. Dojde-li během tvorby systému ke změně dat, dají se datové struktury popsané makry lehce opravit tak, že se oprava nedotkne ostatních programů.

Vytváření uživatelských systémů reálného času s sebou přináší ošetřování velkého množství nestandardních situací a chyb. Ošetřování některých chyb lze algoritmizovat, jiné vyžadují zásah operátora.

Byl vytvořen reentrantní podprogram CHYPRO, s nímž jsou spojeny všechny úlohy, který analyzuje vznikající chyby a zajišťuje jejich výpis na terminál operátora. Používání tohoto způsobu ošetřování chyb značně ulehčuje práci implementátorů uživatelského systému.

V příspěvku jsme se snažili ukázat některé zásady používané při tvorbě systémů reálného času a nastínit řešení, o kterých si myslíme, že jsou nejschůdnější cestou k tvorbě spolehlivých systémů.

Literatura:

/1/ Macháček L., Kudlíková J.: Popis programu SPDLP, interní materiál KS k. ú. o., 1984

/2/ Sokol J., Kaláb P., Štětina I.: Transakční monitory, sborník SOFSEM, 1984

/3/ Šolek J.: Popis podprogramu CHYPRO, interní materiál KS k. ú. o., 1982

/4/ Vaculín Z.: Komunikace operátora s uživatelským systémem pod DOS RV, sborník SMEP '83

/5/ Ježík J.: Soubor makroinstrukcí pro strukturované programování, interní příručka KS k. ú. o., 1982