D I A L E K T I K A A S Ř

(aneb integrovat či rozptylovat?)

Ing. Dušan Streit

Kooperační sdružení zemědělských organizací pro ASŘ a VT
ve Frýdku - Místku

1. Úvod převážně vážně

Dialektika je filozofické učení o nejobecnějších zákonitostech pohybu a vývoje materiálního i duchovního světa, za jejichž vnitřní zdroj se uznává jednota a boj protikladů /1/. Dalšími základními zákony dialektiky jsou přeměny kvantity ve kvalitu a zákon negace negace. Dříve se dialektikou rozumělo umění diskutovat, vést spor. Na tyto tradice bych chtěl svým příspěvkem navázat.

Vzhledem k tomu, co bylo výše uvedeno, bych chtěl, abychom si uvědomili, že současný rozvoj našeho oboru - a to kvantitativní rozvoj - musí zákonitě přerůstat do nových kvalitativních úrovní. Jsem přesvědčen, že spirála vzestupného vývoje se ve výpočetní technice točí tak rychle, že to nemá obdoby; tak rychle, že naše společenské vědomí za tímto trendem zaostává. Tím i zákon negace negace se projevuje tak rychle, že než se podaří jednu kvalitativní úroveň absorbovat, už nadchází druhá, která ji zdánlivě neguje. Nejsme dost pružní na to, abychom se na spirále vývoje nacházeli ve všech souvislostech vždy přibližně na stejné hladině. Tím vznikají zbytečné rozpory navíc a z těch neantagonických "vyrábíme" téměř antagonické. Všem je znám rozpor mezi nabídkou a poptávkou ve výpočetní technice, mezi nedostatkem programovacích kapacit a multiplicitami celých projektů, mezi výkonností procesorů a dostupnými kapacitami vnitřních a hlavně vnějších pamětí, mezi nutností budovat počítačové sítě a nedostatečnými kapacitami spojů apod. Když k těmto rozporům připočteme působící časový nesoulad mezi hardware a software, který mnohdy činí i celou generaci (o zaostávání představ potenciálních uživatelů nemluvě), je situace velmi složitá.

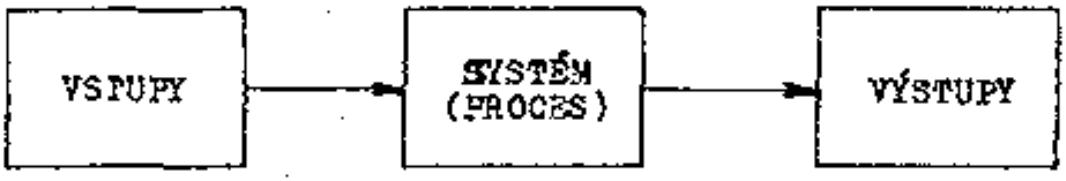
A čas nečeká ... vyjádřeno metaforou: Chceme zastřelit zajíce, který samozřejmě běží. Musíme tedy mířit před zajíce ve směru jeho pohybu. Když "nepředsadíme", zajíc uteče. Vůbec nemáme čas při střelbě shánět lovecký lístek. Nebo devizy na loveckou pušku. "Nekompatibilní" patrony např. z minometu nám taky nejsou nic platné. Když se nám podaří zajíce zastřelit, zjistíme, že zvěřinu vlastně nejíme. Jestli se tato podobnost zdá někomu náhodná, tak věřte, že já už takto "chodím na lov" sedmnáct let. A to pamatuji i doby, kdy se chodilo s kanóny na vrabce. Nyní je zase v módě chodit s personálními pistolkami na slony. A ty potvůrky zajíci jsou v každé generaci rychlejší ...

V poslední době mě zaujal jeden vedlejší projev nového myšlení. A sice "Komplexní program vědeckotechnického pokroku do roku 2000". Všimněte si, ne"rozvoje", ale "pokroku". Cítíte tu změnu od extenzívního k intenzívnímu? I u nás byl rozvoj výpočetní techniky velký. Odpovídá tomuto rozvoji i příslušný pokrok? Zase uvedu příklad. Jistě je dokladem rozvoje státní spořitelny, že přešla na automatizované zpracování vkladů a půjček. Jaký pokrok však cítí uživatel? Že nemůže např. splatit půjčku najednou, ale musí si ještě dvakrát nechat srazit "počítačem" měsíční splátku? To nejsou jenom půjčky, ale i vyúčtování inkasa. Například při ukončení odběru elektřiny není do dvou měsíců provedeno vyúčtování, jak stanoví vyhláška. Zde je z titulu výpočetní techniky porušován i zákon.

Všichni zde víme, že za tyto a podobné zhůvěřilosti nemůže ani počítač ani programátor. Nemůže za to ani dialektika. Každou zákonitost však musíme nejdříve poznat a pak ji uvědoměle využívat. Pokusme se aspoň zamyslet nad některými zásadními otázkami, které můžeme společně ovlivnit. ASŘ je totiž dobrý sluha, ale špatný pán.

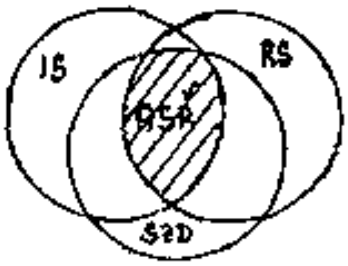
2. ASŘ: systémově či agendově?

Skoro mám strach se na tomto semináři touto otázkou zabývat, aby mě ortodoxní programátoři nevypískali, že tyto úvahy jsou dobré tak pro analytiky nebo jejich šéfy. Pokud zde však nejsme programátoři, živící se pouze programováním her, syntetické hudby a digitalizací obrazu, nemůžeme kontext ASŘ ignorovat; i když uznávám, že vědeckotechnické výpočty a ASŘ technologických procesů má svá vlastní specifika. Konečným kriteriem kvality programu není totiž program sám, ale jeho funkce v systému, který je využíván uživatelem. Uživatel nikdy neocení krásu programátorských triků, tedy vnitřní stavbu programu, ale jeho vnější chování. Navíc většinou pojem program ztotožňuje s úlohou nebo dokonce celým subsystémem. A zde musíme hledat odpověď na naši otázku. Jen systém dokáže skloubit (uspořádat) komplex prvků a jejich vazeb (vztahů) do homogenního celku. Dodávám, že soustave prvků a vazeb je v případě ASŘ hierarchická a může být nazírána z rozlišovacích úrovní včetně příslušných abstrakcí. Dále je ASŘ třeba chápat jako otevřený systém, který má vazby na okolí (vstupy a výstupy), je dynamický (není statický v čase) a vyznačuje se cílovým chováním; takový systém je možno chápat jako transformační proces:



V ASŘ se prolínají vlastně 3 systémy:

- informační systém (IS)
- rozhodovací systém (RS)
- systém zpracování dat (SZD)



K výše uvedenému schematu je třeba dodat, že ASŘ je dán nejen průnikem, ale i sjednocením těchto systémů. Právě na rozdíl od agendového způsobu je třeba pod zorným úhlem automatizace přehodnocovat i obsah IS a RS. Agendový přístup konzervuje stav IS a RS, v jakém byl před budováním ASŘ. Ve výrobě je samozřejmé, že automatizace sebou nese změnu technologie i obsahu všech souvisejících činností. Stejně tak automatizace v administrativě neznamená jen změnu v technologii zpracování dat, ale musí vést k rozbití přežité vazby referent - agenda včetně změn v obsahu činností, ať se to někomu líbí anebo ne. Dělníka se taky nikdo neptá, jestli se přizpůsobí nové technice. Systémový přístup musí všechny tyto vazby postihnout. Následným doplňováním vazeb - tzv. zastřešováním agend (i když jim říkáme subsystémy) - už nikdy systém nevytvoříme.

Dekompozice pak implicitně předpokládá existenci dekomponovaného systému (v našem případě ASŘ). Agenda se dekomponovat nedá, dá se pouze rozdělit (např. mezi 2 referenty). Kriteriem pro dekompozici totiž musí být těsnost vazeb mezi objekty (prvky) tak, aby dílčí části systému (např. subsystémy) měly vnitřní vazby těsnější než vnější vazby. Minimalizujeme tedy vnější vazby a připravujeme podmínky pro bezkonfliktní propojení dílčích částí systému do organického celku. Analogie se strukturovaným a modulárním programováním je nasnadě. U nás bychom tekový přístup nazvali strukturovaným projektováním. Mimochodem si myslím, že pojem "programování" se u nás chápe příliš zúženě, že jeho obsah by měl být širší.

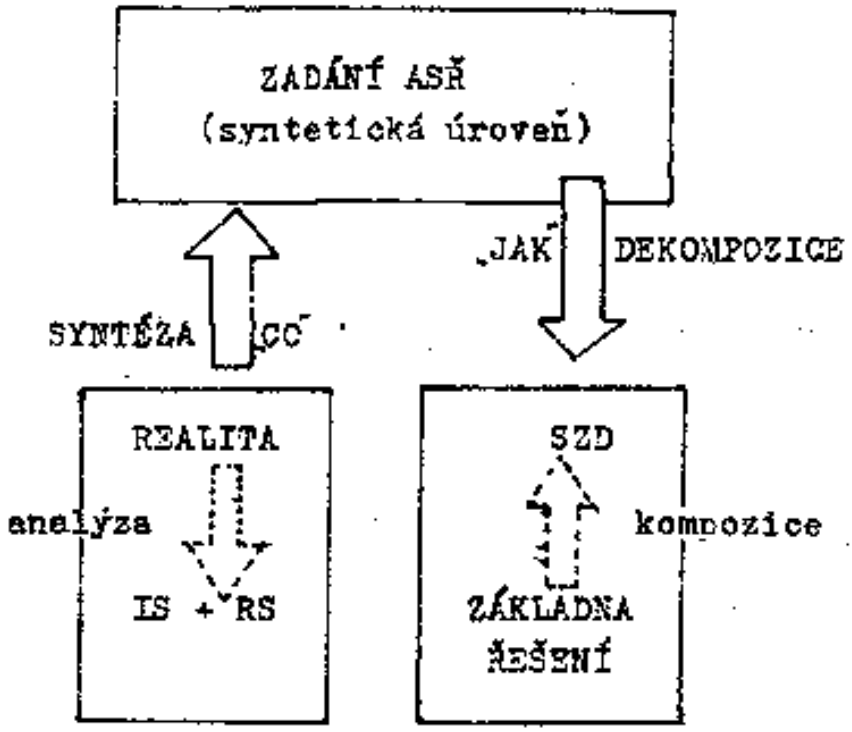
Dialektika není dogma, ale dynamika - nese sebou neustálé konfrontace rozporů směrem k nové kvalitě. Z tohoto pohledu je agenda svou podstatou dogmatická a naproti tomu systém dynamický.

3. Shora dolů či zdola nahoru?

Nechci zde konkurovat literatuře /2/, kde je problematika podrobně rozebrána. Proto budu stručný a omezím se pouze na kontext s předchozím textem.

IS a RS existuje nezávisle na existenci ASŘ. Nemáme prostředky (programové ani technické), abychom jej přímo zadali k řešení na počítači. ASŘ je v tomto smyslu tedy nadstavbou, které umožňuje automatizaci v reálných podmínkách při použití prostředků řešení, o které se můžeme opřít (v terminologii /2/ tzv. základna). Při tvorbě zadání ASŘ (jeho systémové úrovně) převažuje metoda zdola nahoru a jedná se zde o hledisko funkční ("co?"). I když tato etapa v sobě zahrnuje tzv. analýzu současného stavu. Při řešitelské realizaci ASŘ převažuje metoda shora dolů a jedná se o hledisko procedurální ("jak?"). I když v této etapě můžeme povyšovat řešitelskou základnu "prefabrikací" na syntetičtější úroveň jejích prvků.

Znovu se tedy jedná o dialektickou spojitost obou metod, kdy jedna bez druhé nemá význam, jak schematicky znázorňuje nákres.



Levá větev představuje etapu definice ASŘ, pravá větev etapu realizace ASŘ.

4. Obecné či zvláštní programové řešení?

V zásadě lze říci, že obecným řešením je všechno to, co má smysl zahrnout do řešitelské základny. Má to však své hranice v tom, že zahrnutím syntetičtějších prostředků do základny získáme sice unifikovanější prefabrikáty řešení, ale na druhé straně až příliš vázané na konkrétní úlohu. Ad absurdum by to znamenalo konstrukci zvláštního počítače na konkrétní úlohu.

Dialektický vztah vypadá asi takto: zobecňováním získáváme sice syntetičtější, ale zároveň účelovější nástroje řešení. Například vyšší programovací jazyk v sobě zahrnuje oproti assembleru jistá vnitřní zobecnění rozvoje příkazů jazyka do strojového kódu, umožňuje také snadněji dodržovat jisté standardní postupy, ale za cenu jisté vnější specializace jazyka. Ještě vyšší syntetičnost a neprocedurálnost znamená ještě účelovější využití - viz např. parametrické programy, konverzační jazyky apod. Obecný tiskový chod může být obecným pouze proto, že se soustřeďuje na určitý konkrétní problém. Podrobněji jsem tuto problematiku rozvedl v literatuře /3/.

Proto za jediné rozumné kriterium pro vytváření obecných programových řešení považuji technologickou dekompozici systému zpracování dat. Různé zobecňování v konkrétních úlohách a následné přenášení této řešitelské základny do jiných úloh považuji za neefektivní (hovoří se dokonce o přenositelnosti do jiných agend) - srovnej v literatuře /4/.

5. Integrovat či rozptylovat systém?

Dílčí rozpory: a) komplexní - izolované řešení
b) koncentrace - distribuce dat
c) databáze - souborová základna
d) centralizace - decentralizace techn. prostředků
e) nadřazená - autonomní úroveň
f) vzdálené - lokální zpracování
g) spřažené - nespřažené propojení

ad a) Propojováním izolovaných řešení nikdy nemůžeme vytvořit systém, tedy ASŘ, protože došlo k jakési apriorní dekompozici, která nerespektuje vazby v rámci celého systému. Izolované řešení může být opodstatněné pouze tehdy, jestliže se záměrně abstrahujeme od vnějších vazeb, protože jejich automatizace nepřipadá v úvahu, není-li efektivní nebo možná. Při pozdějším případném pohlcení úlohy systémem nelze nový systém budovat zdola od izolovaných řešení, ale je efektivnější od úrovně systému provést novou dekompozici včetně nového řešení úlohy.

ad b) Odpovědět jednoznačně, zda data koncentrovat nebo distribuovat, nelze. Data se nás neptají na místo svého vzniku. Tendence je taková, abychom systém zpracování dat podřídili informačnímu systému a kde je to technicky možné, aby data vstupovala do systému v místě svého vzniku a informace vystupovaly v místě svého užití.

ad c) Jestliže nemůžeme jednoznačně odpovědět na místo naplnění datové základny, měli bychom mít jasno ohledně její definice. Jednoznačně musí být centralizována definice datové základny (popisy, případně adresáře) jako východisko řešení ať už pouze v technickém projektu systému či přímo na počítači. Integrovaná datová základna je jednotícím zdrojem řešení systému. Její důsledné využití vylučuje multiplicity požadavků na vstup dat i redundance v uložení dat. Nemusí se vždy jednat o banky dat, ale jestliže navrhujeme soubory, nemůžeme řešení založit na agendových souborech, jak jsou rozebrány v literatuře /5/. Námety uvádím v literatuře /6/. O výhodách udržování centralizovaných datových definic přímo v počítači není třeba se zvlášť zmiňovat; uplatnění je široké: od dokumentace až po nástroje řešení a nezávislost programů na popisech dat.

ad d) až g) Každý samozřejmě ví, že centralizace, která byla uplatňována v režimu dávkového zpracování a nabyla vrcholu v podobě "closed shop" organizace, ustupuje před požadavkem přiblížit výpočetní techniku uživateli. Bohužel u nás často upadáme do

onečného extrému, kdy na izolované výpočetní technice (minipočítačích, mikropočítačích) provozujeme izolované agendy. Platí zde všechno, co bylo řečeno k bodu a). Navíc otázka propojení přináší ještě komplikaci v podobě případné nekompatibility decentralizovaných zařízení. Mnohdy se tento postup vydává za "z nouze ctnost" a zdůvodňuje se nereálností dálkového přenosu dat a dálkového zadávání úloh. Izolované - byť decentralizované - řešení však má své hranice tam, kde začíná ASŘ. Mám zkušenosti se "zastřešováním" takových "koncepcí" a mohu říci, že jen otázka propojování a konverzí čerpá dvě třetiny programátorských kapacit, o efektivitě nemluvě. Není přitom podstatné, zda na izolovaných a decentralizovaných řešeních chceme vybudovat ASŘ a nebo naopak pod existující ASŘ, který nepočítal s decentralizací, chceme podstavit decentralizované agendy. Zvláště ve druhém případě se nevyhneme duplicitám v požadavcích na vstupy dat, protože centralizovaná definice datové základny nebyla východiskem řešení, jak vyplývá z bodu c). Podstata tedy není v tom, zda techniku decentralizovat nebo ne (to závisí na rozsahu a rozloze organizace), ale zda tato decentralizace byla součástí definice (zadání) ASŘ včetně technického zabezpečení. Tak jako nemůže pro porušování těchto zásad být argumentem otázka relativní nedostupnosti přímého (dálkového) spojení, nemůžeme si dělat alibi ani z relativní nedostupnosti požadovaných technických prostředků. Praktiky typu "ber co je" bez ohledu na celkovou koncepci vede k rozbití ASŘ a mrhání prostředky a kapacitami. Možnostem technického zabezpečení musí být podřízen systém v celku, ne v jednotlivostech a jednotlivých lokalitách. Vyvážené kompromisní řešení je rozumnější než budování dílčích "výkladních skříní" a zanedbání jiných částí systému. Jestliže není reálný dálkový přenos, neznamená to rozbití systému na izolované decentralizované agendy, ale posouzení možností nespřaženého (off-line) propojení. Stejně tak, jestliže není reálné zabezpečit jednotné kompatibilní technické vybavení ve všech decentralizovaných lokalitách, nepostupujeme metodou "každý pes jiná ves", ale posoudíme možnosti, zda otevřený dávkový režim se zkrácenou periodou zpracování nebude dočasným přijatelným kompromisem. Jestliže nám podmínky umožní nejvyšší stupeň decentralizovaného zpracování, tedy on - line počítačové sítě a zpracování

v reálném čase, dbáme na to, abychom co nejvíce odlehčili přenosovým linkám. Tento požadavek předpokládá inteligentní koncová zařízení, která jsou schopna udržovat distribuovanou část báze dat a autonomně ji obhospodařovat. Ovšem se všemi vazbami vzhledem k centrální definici báze dat. Takové řešení předpokládají vhodný databázový a komunikační systém. Nabízí se zde analogie s trendem v hardware, kde již dávno dochází k rozptýlenému řízení, avšak autonomní jednotky tvoří jednotný systém.

Tak tedy integrovat nebo rozptylovat? Abychom mohli hovořit o rozptylování, je nutno nejdříve integrovat, to je nutnou podmínkou. V knize J. Martina "Managing the DB enviroment" je uvedena jako základní podmínka účelné distribuce databáze, aby si rozptýlené databáze zachovaly svou integrační příslušnost k celku; Martin uvádí: "Jsou-li užívány databáze i u koncového uživatele, měly by být tyto databáze a centrální databáze navrhovány ve společném procesu modelování". Odpověď tedy zní: integrovat a rozptylovat.

Je však třeba zachovat určitou posloupnost:

1) Návrh centralizované definice dat, vyřešení banky dat (centrální databáze).
2) Vyřešení komunikačního systému a přenosových cest.
3) Rozptylování báze dat a její využití prostřednictvím decentralizovaných prostředků.

Realizaci je možno zabezpečovat paralelně, nikdy však zdola nahoru.

Ještě poznámka k technickým prostředkům. Nemám rád označení: mikropočítače, minipočítače, střední počítače ... . Toto členění je relativní, v každé generaci bychom museli měnit kriteria co do výkonnosti a kapacity. Navíc mikroprocesory se užívají i u těch největších počítačů. Proto se mi líbí hierarchie: universální, kancelářské (byro), osobní (profesionální) a osobní (soukromé) počítače. V tomto kontextu je zcela zřejmé, že ASŘ jakožto systém vyžaduje řešení využívající na nejvyšší úrovni universální počítače. Málo platné jsou argumenty o výkonnosti dnešních personálních počítačů. Specializované zaměření, odrážející se hlavně v systémových programových prostředcích, je z názvu zcela zřejmé.

6. Závěr (nejen) pro programátory

Ještě jedno dilema nás svužuje při budování ASŘ. A to rozpor mezi programátory a analytiky. I když různými prostředky se snažíme tento rozpor překlenout (RT, HIPO), stále zůstává v organizaci práce a kvalifikaci pracovníků. Byl donedávna dokonce kodifikován v kvalifikačních katalozích. Máme programátory, kteří umějí dobře programovat, i když jim většinou chybí širší odborný rozhled. A na druhé straně máme analytiky, kteří jsou většinou jakýmisi agendovými znalci bez přehledu ve výpočetní technice. Nic proti dělbě práce. Ta ovšem musí být kvalifikovaně řízena. Ne analytikem, protože má vyšší zařazení. A už vůbec ne šéfem - administrativním pracovníkem. Ale "softwarovým inženýrem", který je schopen řídit řešení rozsáhlého systému včetně dělby práce a uplatnění řešitelské modularity, abstrakcí na černé skříňky a jejich vazby. Ve světě jsou tyto metody dostatečně propracovány (viz "programmer chef"). U nás k tomu nevychováváme pracovníky a proto nad technologickou dělbou práce převažuje agendová dělba práce. Mám obavu, že pokud se to nezmění, myšlenky mého příspěvku ortodoxní analytici nepochopí a ortodoxní programátory nebudou zajímat.

Programování v tom širším pojetí je umění kompromisu. Mezi Mezi konstatováním, že to nejde, a že naopak lze vyřešit všechno, je mnoho stupňů. Některé se nedají uspokojivě posoudit v rámci jednoho programu. Nelze například vytvořit absolutně spolehlivé systémy (o tom blíže viz /7/). Vždy je třeba najít správnou míru: co mě to bude stát (práce, prostředků, kapacit) a co za to dostanu (ne já, ale ASŘ).

Příspěvek měl za cíl ukázat, že v našem oboru nelze nic vidět černo - bíle. Nelze např. říci, jedině interaktivní zpracování je správné, dávkové zpracování je přežitek. Velice brzy bychom museli připustit řešení dávkových úloh prostředky pro interaktivní zpracování. Nikde totiž dialektika nefunguje tak provokativně a demonstrativně jako v ASŘ.

7. Závěr (nejen) pro naše šéfy

Tato kapitola by na tento seminář neměla patřit vůbec a jest dost možné, že ji pořadatelé vyřadí, protože jsem již překročil sjednaných 10 stran. Využívám však toho, že seznam literatury k příspěvku patří. Chci pouze upozornit na tyto aspekty:

1) Životnost ASŘ

- je omezená. Jednou se vždy vyčerpají vnitřní zdroje flexibility a je třeba přejít na novou kvalitativní úroveň - nové zadání a realizaci ASŘ.

2) Efektivita ASŘ

- v konečném důsledku se vždy musí realizovat uvnitř některé organizace, v její výrobě, administrativě. Organizace by měla rozhodnout, jestli je konečným spotřebitelem nebo dodavatelem. Směšovat tyto dvě stránky není dialektika. Nelze chtít, aby se VS zaplatilo službami pro cizí a zároveň "zčarma" zavádělo vlastní ASŘ.

3) Úloha ASŘ

- není v tom, že se "šetří lidi". Naopak stimuluje novou poptávku po kvalifikaci některých profesí. ASŘ neautomatizuje pouze to, co by zvládli lidé, ale řeší především to, co bez VT není řešitelné. Kdysi jsme byli svědky takových přímětů, že počítač zvládne to, co by 10 000 matematiků řešilo 100 let. A stále jsme žádného matematika neušetřili, právě naopak.

L I T E R A T U R A

/1/ Ilustrovaný encyklopedický slovník, Academia, 1982

/2/ Lexa I.: Shora - dolů nebo zdola - nahoru. Sborník Programování 87, DT ČSVTS Ostrava, 1987

/3/ Streit D.: K problematice parametrických programů. MAA 7/81

/4/ Vojáček V., Pechlát J.: Je parametrický program efektivní? Sborník Programování 82, DT ČSVTS Ostrava, 1982

/5/ Chvalovský V.: Banky dat. SNTL, 1984

/6/ Streit D.: Integrovaná informační základna. Sborník Programování 82, DT ČSVTS Ostrava, 1982

/7/ Streit D.: Provozní spolehlivost aplikačního software. Sborník Programování 85, DT ČSVTS Ostrava, 1985