APS - profesionální LOWER CASE pro počítače architektury IBM SAA

Vladimír Zyka

1. Úvod

APS představuje komplexní nástroj pro design, programovou realizaci, dokumentaci a údržbu rozsáhlých informačních systémů, implementovaných na počítačích spadajících do architektury IBM SAA, tzn. od IBM PC s operačním systémem MS-DOS (PCDOS) nebo silná PC s operačním systémem OS/2 popř. Windows, přes midrangeové počítače IBM AS/400 až k mainframeovým počítačům IBM 4361.

Tímto nástrojem je možno vytvářet jednak jednodušší aplikační programová vybavení pro jednu z výše uvedených hardwareových platforem např. programové vybavení pro počítače PC nebo rozsáhlé informační systémy přesahující homogenitu hardware např. informační systémy s distribuovanou datovou základnou (pracovní stanice s Presentation Managerem či Windows a lokální databází komunikující s jedním nebo více servery (silné PC s OS/2, AS/400 či maniframes).

Pro každou z uvedených hardwareových platforem je možno použít k implementaci programového vybavení různých typů základního programového vybavení.

Pro počítače PC s operačním systémem OS/2 je možno použít:

- pro tvorbu komunikační vrstvy (user interface) následující základní programová vybavení:
  - Presentation manager,
  - IBM CICS for OS/2,
  - Micro Focus CICS,
  - Micro Focus IMS
- pro manipulaci s daty tzv. VSAM organizaci indexních nebo sekevenčních souborů nebo následující profesionální SQL databáze:
  - IBM DB2/2,
  - ORACLE,
  - SYBASE,
  - XDB.

Pro počítače PC s operačním systémem Windows lze použít:

- pro tvorbu komunikační vrstvy:
  - Windows,
  - Micro Focus CICS,
  - Micro Focus IMS,
- pro manipulaci s daty tzv. VSAM organizaci indexních nebo sekvenčních souborů nebo následující SQL databáze:
  - XDB,
  - Microsoft SQL Server.

Pro počítače řady IBM AS/400 je možno použít pro tvorbu uživatelského rozhraní systému DDS, jenž umožňuje vytvářet znakové uživatelské rohraní pro AS/400 a je součástí jádra operačního systému OS/400. Pro manipulaci s daty na IBM AS/400 je možno použít volitelně buď systém DDS (indexní souborová organizace dostupná z 3GL jazyků např. COBOLu pomocí standardních příkazů OPEN, CLOSE, READ, WRITE, REWRITE, DELETE atd.) nebo generaci kódu s prekompilací do SQL/400.

Pro mainframeové počítače je možno generovat kód do následujících produktů:

- IDMS
- CICS
- DB2
- ISPF - VSAM
- IMS

Při vytváření rozsáhlých informačních systémů lze použít architekturu client/server, kde clientem může být silné PC s GUI pod Presentaion Managerem nebo Windows komunikujícím s dalšími programy (serverovými) buď na serveru PC s operačním systémem OS/2 nebo s mainframeovým počítačem. V takovémto případě se generouje kód s API funkcemi tzv. APPC a APS automaticky vytváří všechny součásti programu (tzn. jednak program pro OS/2 či Windows tak i programy pro manipulaci s daty na OS/2 serveru nebo na mainframeu). Je-li při generování kódu zvolen cleint s OS/2 s databází DB2/2 je možno pomocí základního programového vybavení DDCS/2 komunikovat s různými databázemi:

- s SQL/400,
- s DB2 (mainframe),
- s DB2/6000 (IBM RS/6000) nebo
- s dalším DB2/2.

Z výše uvedeného výčtu možností je evidentní, že APS umožňuje pohodlně implementovat jakýkoliv informační systém na libovolnou hardwareovou platformu zapadající do IBM SAA.

Při použití DDCS/2 je možno velice pohodlně implementovat informační systémy s ditribuovanou datovou základnou na různých počítačích IBM i když se nejedná o klasickou architekturu client/server realizovanou pomocí API do APPC. V současné době však DB2/2, DB/6000, SQL/400 a DB2 spojené pomocí DDCS/2 představuje nejprogresivnější počin IBM. DDCS/2 s protokolem DRDA zajišťuje integraci zpracování dat v architektuře IBM SAA a IBM tuto integraci velice silně podporuje. V současné době již existuje celá řada vývojových nástrojů (generátorů, LOWER-CASE), jenž tuto architekturu podporují a to i přímo od IBM. Na letošním veletruhu CEBIT 94 představila IBM kromě obrovské propagace OS/2 také GUI nadstavbu pro OS/2 právě nad DDCS/2, jenž je obdobou Query manageru, ale narozdíl od něj umožňuje zpracovávat pomocí neprocedurálních SQL příkazů data ve všech databázích komunikujících s DDCS/2.

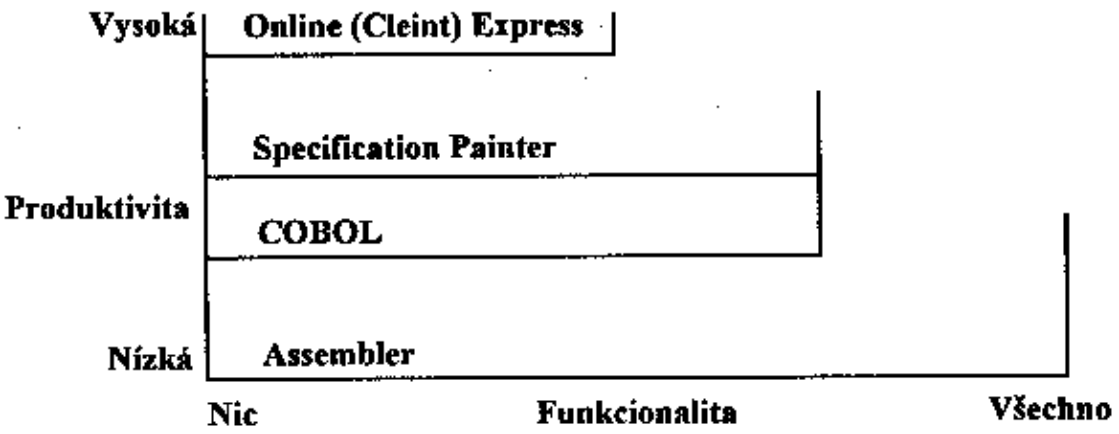
2. Vnitřní struktura APS

APS není pouhý generátor kódu. Jedná se o komplexní nástroj velice úzce spolupracujici s UPPER CASE Excellerator - II či ADW of firmy Knowledgeware. Jádrem tohoto nástroje je smozrejmě velice výkonný a rozsáhlý generátor programů v jazyce COBOL. Jádro generátoru pochází z dob slávy mainframeů. Jeho neustálým zdokonalováním a rozšiřováním jeho základních funkcí docházelo k jeho portabilitě na různé typy hardwareových platforem. Zdrojovým tvarem

jednotlivých navrhovaných funkcí je tzv. S-COBOL (Super COBOL) vycházející jednoznačně z COBOLu. Jádro generátoru obsahuje makroprocessor, který umožňuje uživateli definovat výkokmné příkazy (složenné z příkazů COBOLU či dalších maker). Díky tomuto makroprocesoru (APS Customization facility) je generátor neustáie vývojově otevřený a umožňuje do budoucnosti vytvářet další a další systémová či uživatelská makra, což ve svém důsledku umožňuje zajišťovat další portabilitu tohoto nástroje a další zvyšování efektivnosti v generování kódu.

S-COBOL je v podstatě 4GL jehož jádrem je COBOL. Tento jazyk je složen jednak ze systémových maker, jenž jsou součásti tohoto produktu, jednak z uživatelem definovaných maker. Další součástí tohoto 4GL je kompletní sada klíčových slov COBOLu. Znamená to tedy, že je možno pomocí APS vytvářet stejně složité programy jako v COBOLu. Důležitou odchylkou v psaní programu pomocí tohoto 4GL je skutečnost že, u každého příkazu je možno definovat tzv. klíčové slovo (keyword) jenž generátoru říká, do kterých sekcí danou část vygenerovat. Zaniká tedy nutnost rozdělovat program do základních čtyřech divizí. Pracovní proměnné je možné např. uvádět mezi vákonné příkazy atd.

Obr. 1.1 Závislost funkcionality na produktivitě jednotlivých úrovní vývojového programového vybavení



APS je z uživatelského hlediska rozčleněno do několika dílčích nástrojů pro návrh programů a do několika desítek dalších pomocnýh funkcí.

2.1. Aplication Painter (AP)

AP slouží k základní kompozici dané funkce. Umožňuje definovat dílčí funkce, jenž jsou představovány jednotlivými programy. K jednotivým programům definovat obrazovky, okna, sestavy, pracovaní datové struktury a tzv. subschémata (parciální pohledy aplikací či programů na datovou základnu). Z AP je možno spouštět další nástroje pro vývoj.

2.2. Screen Painter (SP)

SP slouží k rychlému designu znakové obrazovky. Návrh obrazovky spočívá v zakreslení textů a vstupních, vstupně-výstupních, výstupních polí. Při definici polí je možno využít informace z

jednotlivých polí datové základny (odvození základních atributů tohoto pole z datové základny - typ, velikost, tzv. field edits nebo definovat nová pole. Ke každému poli lze definovat další atributy zobrazení dle typu základního aplikačního vybavení (např. pro DDS je možno použít tzv. indikátory umožňující dynamicky v programu následně měnit vlastnosti tohoto pole).

2.2.1. Field edits

Pro každé obrazovkové pole je možno definovat jeho vnitřní reprezentaci (typ, rozměr), vstupní editace, výstupní editace, uvést výčet platných hodnot pro toto pole, definovat chybová hlášení v případě chybného vyplnění či nevyplnění, spustit libovonou aplikační rutinu, kterou si sami navrhnete. Aplikační rutiny mohou být navrženy buď formou podprogramu nebo formou lokálního kódu (neopakovatelné pro další použití) nebo spustěním podfunkce navržené v téže funkci v jiném kontextu (Perform) nebo použitím globálního kódu (podfunkce již je možno zařazovat do dalších programů) nebo definováním uživatelského makra.

Pomocí aplikačních rutin lze provádět kontextové nápovědy a nestandardní kontrolní mechanismy pro vstup dat v obrazovkových polích, v nichž je třeba např. zajistit konzistenci s atributy uvedenými v jiných entitách (simulace forign key).

2.2.2. User help

Součástí SP je nástroj na definici uživatelské příručky. Nápovědu lze definovat na několika úrovních:

- úroveň aplikace - celkový popis aplikace, přehled funkcí aplikace,
- úroveň funkce - popis dílčí funkce,
- úroveň obrazovky - popis funkce z hlediska jedné obrazovky, popis významu polí a způsob ovládání obrazovky,
- úroveň pole - význam pole popř. způsob zadávání hodnot do tohoto pole (kontextová nápověda je součástí tzv. Field edits).

2.3. GUI painter

Jedná se o plně grafický nástroj pro tvorbu grafických oken obsahující vstupně-výstupní pole, statické texty, listboxy, comboboxy, multiple line fields. Pro každé okno je možno definovat několikaúrovňové pop-down menu. Lze ovládat plně barvy, fonty, pozadí, hilight, reverse pomocí tzv. Presentation parameters. Taktéž je možno na obrazovce definovat tzv. bitmapy, icony, pointery. Jednotlivé prvky okna mohou být vrstveny přes sebe tzn. lze vytvořit pozadí z digitalizovaného obrázku a na něj umístit další texty či vstupně-výstupní pole. GUI painter umožňuje dále definovat několik základních typů pro MENU. Základním typem MENU je závěsné pop-down MENU.

Další typy MENU:

- ICON - jednotlivé volby jsou představovány iconami,
- PUSHBUTTONS - možno označovat funkce pomocí tvz. tlačítek,
- RADIO BUTTONS
- THREE STATE BUTTONS.

Při definici vstupně-výstupních polí lze využít taktéž odvození základních atributů pole z vlastností specifikovaných v datové základně. Ke každému takovému poli lze definovat taktéž Field edits.

2.4. Mockup painter

Pomocí tohoto nástroje je možno definovat základní struktury výstupu přímo na obrazovku. Pro každou funkci představující tiskovou sestavu se pomocí tohoto nástroje definuje na obrazovce jednak rozložení sestavy (konstatní názvy, záhlaví, součtové řádky, jména polí, mezisoučty, číslování stránek) jednak rozmístnění výstupních polí a jejich mapování s adekvátními poli z datové základny či pracovními poli definovanými buď pomocí Data structures painteru nebo přímo v Specification painteru.

Nástroj je vhodný pro tvorbu hromadných výstupů, skupinových algoritmů. Nelze pomocí něj generovat grafické výstupy např. doklady či jiné složité grafické výstupy.

V APS neexistuje žádný nástroj pro tisk grafických výstupů. Absence tohoto nástroje znamená pro APS největší handicap proti ostatním generátorům či LOWER CASEs.

2.5. Data structures painter

Tento nástroj slouží k definici všech pracovních proměnných, jenž jsou použity v generovaných programech. Zde se deklarují taková vstupně-výstupní pole, jenž nemají přímou reprezentaci v datové základně. Jedná se tedy o odvozené dataelementy, výběrová pole, agregovaná pole atd. Datové struktury mohou obsahovat jednu nebo více struktur složených z jedné nebo více proměnných. Data structures je možno přiřazovat k celé aplikaci nebo pouze k jedné funkci či podfunkci. Z hlediska přehlednosti je ideální rozčlenit proměnné do dvou skupin:

- dataelementy společné pro celou aplikaci
- dataelmenty potřebné pro jednu funkci

Je možno deklarovat další data structures přímo v jednotlivých lokálních podfunkcích či makrech avšak z hlediska následné údržby je tento způsob nevhodný, neboť se ztrácí centrální přehled o těchto proměnných.

2.6. Specification painter (SpecP)

SpecP umožňuje vytvářet interaktivní nebo batchové programy, které reprezentují jednotlivé funkce z dané aplikace. Programy vytvářené pomocí SpecP jsou složeny a využívají výstupy všech předchozích nástrojů a pomocí 4GL S-COBOL spojují všechny předchozí specifikace funkcí a podfunkcí. Teoreticky je možno ve SpecP vytvořit samostatně stojící program, jenž je složen pouze z Cobolských příkazů a tzv. keywords (informací o tom, v kterých divizích a sekcích se má příslušný příkaz sestavit). Jedná se samozřejmě o teoretickou konstrukci dokreslující možnosti APS. Není-li možno např. řešit zcela atypickou funkci pomocí žádného z výše uvedených nástrojů a nestačí-li k návrhu takovéto funkce ani žádný příkaz či sada příkazů 4GL S-COBOL, je možno v SpecP vyřešit tuto funkci přímo napsáním "čistého" Cobolského programu. Lze tedy konstatovat, že co "umí" kompilátor Cobolu umí i APS. APS je v podstatě silná a komplexní nadstavba na jazykem COBOL s celou řadou podpůrných funkcí sloužících k designu, kódování a údržbě programového vybavení. Z hlediska víceplatformového nasazení se jedná o komplexní nadstavbu nad různými kompilátory Cobolu počínaje kompilátorem od firmy Micro Focus na PC přes COBOL/400 až ke kompilátorům na mainframes.

2.7. Scenario painter (ScenP)

ScenP umožňuje vytvářet tzv. prototiping. Úzce souvisí s GUI painterem nebo Screen Painterem a umožňuje vytvářet sekvence obrazovek tak, jak se budou uživateli ukazovat aktivací jednotlivých funkcí aplikace. ScenP umožnuje, aby jednotlivé obrazovky či okna byly vyplňovány testovacími hodnotami a tyto hodnoty je možno v plánovaných sekvencích přenášet na následující obrazovky.

Tento nástroj umožňuje tzv. metodu RAD (Rapid aplication development) viz dále.
Sofwareový designér může přímo s uživateli kreslit obrazovky či okna a řadit je za sebe. Tato metoda často vytváří u uživatelů úžas nad rychlostí projekce, neboť spuštění scenaria vytváří i u zkušených uživatelů iluzi o hotovém produktu. Obrazovky, které se používají v ScenP je samozřejmě možno použít bez dalších úprav i pro programovou realizaci funkce.

2.8. Client express (OX)

OX je nejsilnějším a nejprograsivnějším nástrojem v APS. V podstatě se jedná o grafickou nadstavbu nad SpecP, umožňující velice produktivně a velice formalizovaně navrhovat jednotlivé funkce aplikace. OX je nástroj, v němž pomocí různých nabídek, MENU, ICON a PUSHBUTTONS i méně zdatný programátor velice rychle a velice formalizovaně specifikuje problematiku navrhované funkce.

OX autonomně vytváří základní skeleton programu. Na základě typu obrazovky či okna vytváří tři základní typy skeletonu:

- skeleton pro obrazovku, jenž umožňuje zobrazovat, opravovat, vkládat nebo rušit jeden výskyt entity, složené i z více fyzických tabulek či souborů,
- skeleton pro obrazovku zobrazující seznam,
- skleten představující kombinaci typu a) a b).

V každém skeletonu jsou designérovi dostupná určitá kritická místa, v nichž specifikuje buď podfunkci formou lokálního kódu, globálního kódu, spuštěním podfunkce téhož programu (PERFORM), podprogramem, systémovou rutinou a makrem nebo provádí customizaci navrženého řešení dané systémové rutiny.

Při návrhu obrazovky je vhodné pojmenovávat jednotlivé MENU funkce či PUSBUTTONs (např. závěsné menu Konec práce názvem KONEC, oprava výskytu entity jménem OPRAVA atd.). V opačném případě APS automaticky generuje názvy pro takovéto objekty, což má za následek nižší vypovídací schopnost a přehlednost.

V OX se automaticky objevují všechny objekty specifikované na obrazovce či oknu a jakákoliv aktivita s jakýmkoliv objekem je chápána v OX jako událost vyvolávající určitou podfunkci, kterou může designér navrhnou či nikoliv. Nenavrhne-li designér pro některý z objektů žádnou podfunci, pak se aktivací tohoto objektu nic neprovádí. V opačném případě se provádí designerem specifikovaná podfunkce. V navrhované funkci se pracuje s různými typy objektů. Každý typ objektů je charakterizován specifickými vlastnostmi a tím pádem reaguje na různé události různě.

OX má pro typy objektů pojmenované stavy, v nichž se objekt může nacházet. Každý takovýto stav je představován kritickým místem funkce, do nějž může designér včlenit podfunkci.

Podfunkce může být představována:

- lokálním kódem - k dané události se otevře editor, do něhož je možno funkci specifikovat pomocí 4GL S-COBOL,
- globálním kódem - v aplikaci vždy existuje určitá množina identických podfunkcí, záleží pouze na podrobnosti analýzy a návrhu dané aplikace a na vyspecifikování takovýchto podfunkcí, výhodou

použití globálního kódu je kromě úspory času v opakovaném psaní stejného kódu i zvýšení formalizovanosti zápisu funkce a tudíž i úspora času v případě změny logiky v takovýchto elementárních podfunkcích (v takovém případě stačí pouze změnit pomocí SpecP tento globální kód a přegenerovat celou aplikaci,

- spuštěním podfunkce téhož programu (PERFORM)
- systémovou rutinou - OX generuje na základě specifikace tzv. DATABASE ACCESSes základní konstrukci pro zabezpečení specifikovaného DATABASE ACCESS (standardním způsobem naplňuje výběrové klíče, provádí standardním způsobem vyhledání dat v datové základně, standardním způsobem provádí ošetřování nestandardních stavů - konec souboru, neexistence záznamu, duplicitní klíče atd), obdobným způsobem generuje rutiny pro opravu, založení či zrušení entity popř. sloučení entit). Tyto předpřipravené systémové rutiny obsahují opět určitá kritická místa, v nichž může designér provádět detailní specifikaci či vlastní ošetřování nestandardních stavů tzv. customization,
- uvedením uživatelského makra, jenž v okamžiku generace na základě generačních parametrů rozbalí makro v sled příkazů
- podprogramem - lze spustit samostatně navrženou funkci aplikace.

Dalším krokem v definici funkce pomocí OX je tzv. Field Mapping. Tato funkce představuje spojování vstupně-výstupních polí obrazovky dané funkce s konkrétními dataelementy z datové základny popř. pracovními proměnnými funkce navrženými pomocí Data Structures Painteru. Obrazovková pole lze mapovat jako:

- vstupní,
- obousměrná
- výstupní.

Tato operace zajistí po každém přečtení obrazovky programem přesun obsahu obrazovkových polí do určených databázových dataelementů či do specifikovaných pracovních proměnných. Obdobně před zobrazením obrazovky či okna provede přesun hodnot adekvátních databázových elementů či pracovních proměnných do obrazovkových polí.

Nejsilnější součástí OX je definice tzv. DATABASE ACCESS. Tato funkce je představována opět sadou ICON, PUSHBUTTONS a MENU, pomocí nichž lze definovat i ty nejsložitější kombinace přístupů k jednotlivým tabulkám či souborům. Nejsložitější DATABASE ACCESS v jedné funkci může být složen až ze 12 dílčích DATABASE ACCESSes. Elementární DATABASE ACCESS může být tvořen JOINem až 15 tabulek. Znamená to tedy teoreticky, že data na jedné obrazovce či okně mohou být uloženy ve 180 tabulkách relační databáze.

Designér pomocí DATABASE ACCESS definuje logiku přístupu k datům datové základny. Nejprve si stanoví postup při vyhledávání dataelementů, které na obrazovce zobrazuje popř. ve funkci zpracovává (jednoduchým způsobem specifikuje tabulky databáze či soubory v optimálním pořadí, v nichž jsou potřebné dataelementy obsaženy). Poté definuje pro každou vyspecifikovanou tabulku či soubor typ operace, kterou hodlá s danou tabulkou provádět (načtení, oprava, založení, zrušení - může uvést jen jednu nebo i více typů operací). Dále definuje, s kterými dataelementy ze specifikované tabulky bude pracovat (z největší pravděpodobností nebude potřebovat zpřístupnit všechny dataelementy - atributy z každé tabulky). Specifikuje-li designér tyto dataelementy optimalizuje tím programovou realizaci (zvyšuje její spolehlivost a průchodnost, neboť při použití relačních databází dochází k předávání pouze specifikovaných dataelementů, čímž se snižuje objem přenášených dat mezi správcem databáze a programem a tím se v síťové verzi snižuje i objem přenášených dat po linkách). Designér však tuto specifikaci nemusí provádět. Dalším krokem ve specifikování elementárního DATABASE ACCESS je tzv. Qualifications, pomocí něhož designér definuje podmínky, podle nichž se provádí specifikované typy operací. Součástí modelování elementárního DATABASE ACCESS jsou další možnosti vyplývající např. ze syntaxe SQL příkazu (GROUP BY, skalární funkce, HAVING, ORDER BY atd.).

Na základě výše uvedených specifikací elementárního DATABASE ACCESS je schopen OX standardním způsobem realizovat takovouto operaci. V každém elementárním DATABASE ACCESS existují opět kritická místa, jenž jsou z OX dostupná a je tudíž možno, aby do těchto míst, v případě potřeby, provedl designér dospecifikaci podfunkcí či pozměnění standardního ošetření nestandardních stavů.

Vyspecifikované elementární DATABASE ACCESSes jsou ve funkci dostupné pomocí tzv. systémových rutin *QUERY, *ADD, *DELETE, *UPDATE. Takovéto systémové rutiny je možno aktivovat jako reakci na libovolnou událost libovolného objektu funkce či v kterémkoliv kritickém místě funkce a to přímo jejich označením (např.
na obrazovce či okně bude navržena funkce Načtení informací z datové základny na obrazovku a tato funkce bude z hlediska designéra označena slovem NACTI, pak v OX bude figurovat jméno NACTI, ke kterému designér specifikuje funkci *QUERY, čímž zajistí spuštění rutiny zabezpečující i velice složitý a propracovaný DATABASE ACCESS).

Specifickým typem operace s tabulkou či souborem je tzv. LOOP, jenž zabezpečuje načtení seznamu výskytů entity či JOINu entit buď do LISTBOXu v okně nebo do opakujícího se pole na obrazovce a logiku související s pohybem dopředu nebo dozadu v takovémto seznamu.

Podle propagačních materiálů fy INTERSOLV lze pomocí OX realizovat přibližně 80 procent všech funkcí navrhovaného inforamčního systému, přičemž OX zvyšuje produktivitu oproti SpecP přibližně o 30 procent. Jeho podstatnou výhodou oproti použití SpecP je dokonalá struktura a striktní formalizovanost popisu funkce, což přináší největší efekt v etapě údržby programového systému.

Podle zkušeností projektantů a programátorů společnosti ITS a.s. jsou tyto odhady parametrů kvalifikované. ITS realizoval rozsáhlý informační systém obchodní organizace (cca 600 zaměstnanců, tisíce zakázek, desetitisíce faktur za rok) představující přibližně 600 programů, přičemž pouze přibližně 20 programů nebylo realizováno pomocí OX (byly realizovány pomocí SpecP).

2.8. Report

APS obsahuje zároveeň komponentu umožňující získávat tiskové výstupy o jednotlivých objektech aplikace. Je možno použít celou paletu různých sestav obsahujících přehledně uspořádané objekty aplikace, jejich vzájemné vazby popř. velice důležité vzájemné vazby mezi těmito objekty.

Tato část APS slouží zejména k dokonalé programové dokumentaci aplikace popř. k rychlému získání informací potřebných pro provedení určité systémové změny dotýkající se celé aplikace.

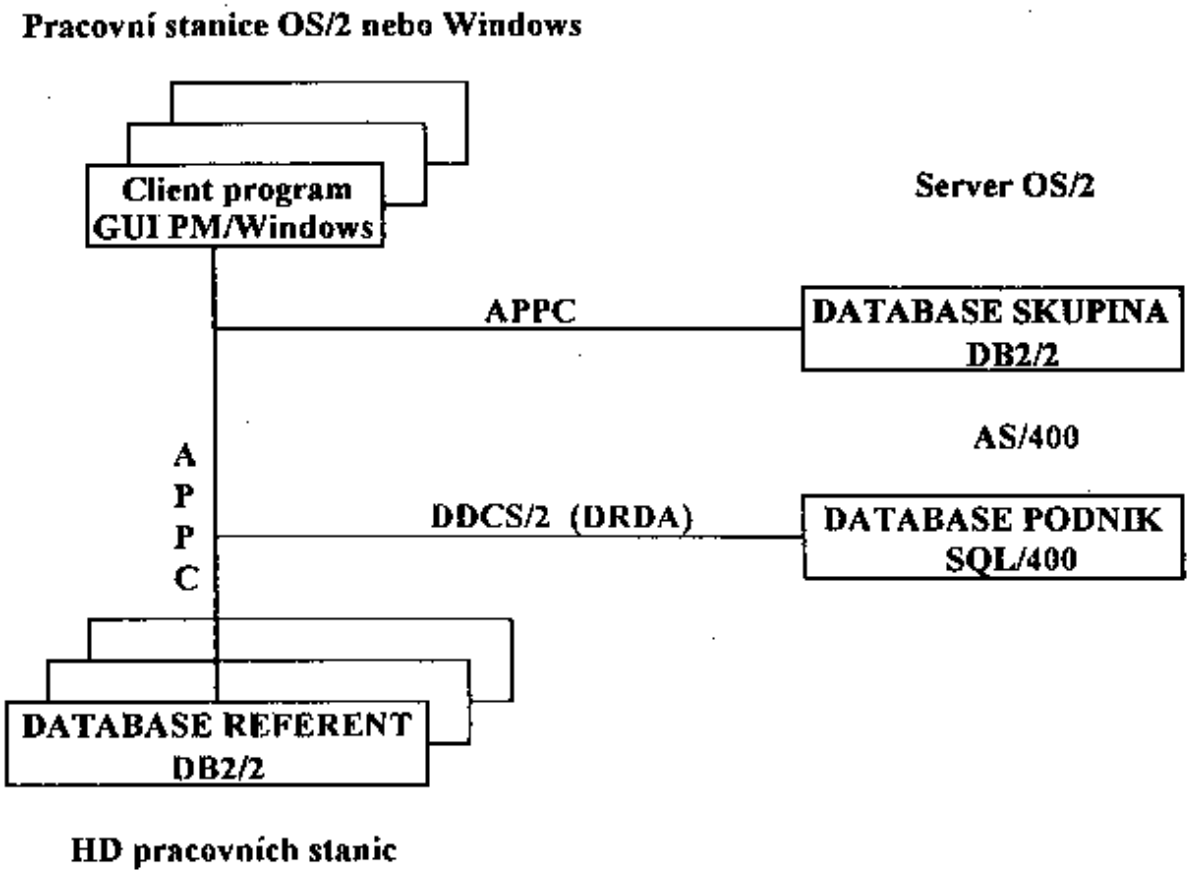
2.9. Realizace architektury client-server

APS umožňuje velice pohodně zabezpečit programovou realizaci v architektuře client-server.

Jednotlivé fyzické tabulky popř. view navržené a optimalizované datové základny (např. pomocí systému pro podporu analýzy Excellerator - II) lze rozdělit do několika skupin. Kritériem členění je úroveň a technologie zpracování daných entit. Každé skupině takto rozčleněných tabulek pak může odpovídat samostatná relační databáze. Jednotlivé relační databáze jsou distribuovány na různém HW (např. na serveru OS/2 či na AS/400). Pro každou funkci aplikace (program) je pak nutno specifikovat jména dílčích programů a označit v APS na jakém typu HW bude dílčí program spouštěn. APS automaticky vytvoří jednotlivé dílčí programy (automaticky pozná, z kolika dílčích serverových programů se skládá a nageneruje do jednotlivých dílčích programů vstupně-výstupní operace pro použité databázové tabulky dle jednotlivých skupin tabulek). Tyto nagenerované dílčí programy obsahují API funkce APPC.Pomocí APPC tyto dílčí programy autonomě komunikují a spojují jednotlivé databáze.

Obr.: 2.1. Příklad architektury client-server.

Jednotlivé funkce (programy) mohou být představovány v tomto případě čtyřmi dílčími programy vzájemně spolu komunikujícími spuštěnými na třech různých počítačích.



3. Další provozní funkce APS

APS obsahuje desítky provozních funkcí zajišťujících integritu vývojového prostředí s ostatním základním vývojovým programovým vybavením. Z prostředí APS je možno souštět snadným způsobem např. vývojové prostředí Micro Focus Workbench, Query Manager DB2/2 atd.

Další skupinu provozních funkcí tvoří funkce umožňující implementovat navržené řešení do zvoleného prostředí (např. spouštění LINK programu vytvářejícího z jednotlivých programů samostatně spustitelný program .EXE nebo generování .BND souborů sloužících k připojování aplikačních programů k skutečným fyzickým databázím v DB2/2 atd.).

Velice významnou podmnožinou provozních funkcí APS jsou funkce zajišťující integritu APS s dalšími CASE nástroji. Na vysoké úrovni je propracován interface do upper CASE Excellerator II. APS spolupracuje s dalšími upper CASEes (např. ADW od firmy Knowledgeware). Do APS je integrován i produkt sloužící pro řízení údržby projektu tzv. Intersolv PVCS umožňující definovat všechny objekty pro jednotlivé verze projektu, projekt uzavírat, odvozovat z něj další projekty popř. zdokonalovat některou z uzavřených verzí (podrobnosti o tomto produktu podává plánovaný příspěvek od společnosti AIT spol s r. o. Praha).

4. Zkušenosti s použitím APS v etapě vývoje rozsáhlého informačního systému.

Společnost ITS a.s. realizovala v druhé polovině roku 1993 a v prvních měsících roku 1994 rozsáhlý informační systém velké zahraničně obchodní organizace (cca 600 zaměstnanců, poměrně vyspělá organizační struktura podniku - obchodní a neobchodní útvary, řádově tisíce obchodních případů za rok).

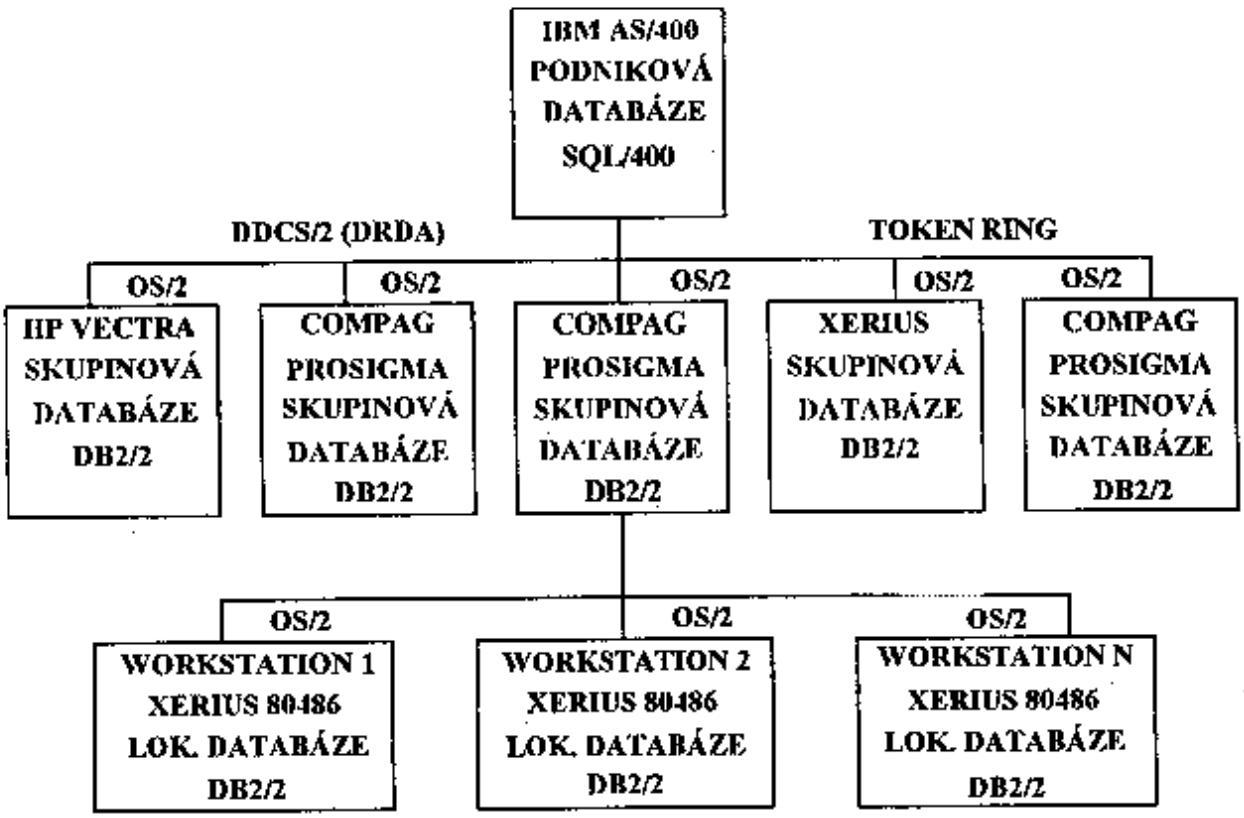
Analytické a projektční práce na projektu jsme prováděli na základě metody RAD (Rapid aplication development) doporučované firmou Intersolv.

Pro analýzu a návrh struktury datové základny jsme používali upper CASE Excellerator II. Současně s analýzou jsme prováděli návrh uživatelského interface, který jsme demonstrovali a konzultovali se zástupci z řad uživatelů a pružně jsme tak zapracovávali jejich požadavky a potřeby do budovaného projektu. Uživatelský interface jsme navrhovali pomocí APS (GUI painteru, Scenario Painteru). Takto prováděný prototyping má kromě čistě pragmatických výhod i výhodu, že vtahuje koncové uživatele velice aktivně do procesu návrhu projektu.

4.1. Technologický popis projektu

Projekt obsahuje nyní přibližně 600 programových jednotek, 170 grafických oken, 30 negrafických tiskových funkcí, 5 grafických tiskových funkcí. Datová základna je rozčleněna do třech databází a tyto databáze obsahují přes 150 tabulek. Programové vybavení je provozováno v heterogenní síti počítačů.

Obr.: 4.1. Sktuktura počítačové sítě Skloexport a.s.



Datová základna projektu je distribuovaná do třech úrovní (Podnik, Skupina, Referent). Jednotlivé funkce jsou realizovány architekturou client-server. Toto rozložení dat má příznivý dopad na rovnoměrné vytížení zdrojů jednotlivých komponent sítě a na minimalizaci přenášených informací po fyzických linkách sítě. I přesto byl pro propojení skupinových serverů s AS/400 vybrán velice výkonný a spolehlivý Token Ring.

V síti je pomocí Token Ring propojeno šest serverů. Ke každému serveru je pomocí Ethernet připojeno průměrně 15 pracovních stanic. Ke každému skupinovém serveru je připojena jedna kvalitní a výkonná laserová tiskárna pro zajištění grafických výstupů (dokladů - faktur, expedičních příkazů atd.) a výkonná řádková (24 jehličková) tiskárna určená pro tisky sestav a přehledů.

Datová základna je implementována do databáze IBM DB2/2, jenž je pomocí DDCS/2 integrována s databází SQL/400. Uživatelské rozhraní je plně grafické pod Presentation Managerem. Sdílení zdrojů v síti (PEER-TO-PEER) je zajištěno pomocí IBM LAN SERVER 3.0. Správa komunikací (emulace terminálu 5250 - terminál AS/400 na každém serveru a na všech pracovních stanicích, správa programů client-server) zajišťuje IBM Communications Manager/2. Grafické výstupy jsou realizovány pomocí textového editoru AmiPro for OS/2 popř. AmiPro for Windows.

4.2. Závěrečné zhodnocení APS v procesu vývoje a implementace projektů.

APS je efektivní používat při tvorbě rozsáhlých programových vybavení. Pomocí APS nemá smysl vytvářet malé projekty. APS generuje pouze do profesionálních prostředí (DB2/2, ORACLE atd.). APS nelze v žádném ohledu porovnávat s PC nástroji (FoxPro, Clipper atd.).
APS generuje programové vybavení, pro jehož spuštění není třeba žádný "run time".

Portabilita programového vybavení je uvedena v kapitole 1. Pomocí APS lze využít naprosto všechny vlastnosti zvoleného implementačního prostředí. Díky této vlastnosti lze vytvářet skutečně profesionální a dokonalá programová vybavení ve své kategorii. Za tuto výjimečnou vlastnost však APS platí vysokou daň v následné portabilitě již realizovaného programového vybavení. Je-li v daném řešení použito specifických funkcí a možností původního implementačního prostředí, potom přechod takovéhoto řešení do jiného prostředí je velice komplikované a je v určitých případech nutno tyto specifické funce nahradit jinými prostředky. Tuto vlastnost nelze u APS chápat jako nevýhodu, ale jako výhodu. Vývojové prostředky, jenž vykazují vysokou úroveň portability řešení (PROGRSS, vývojové produkty Micro Focus Dialog System, AAI a Micro Focus Workbench a jiné) neumožňují využít opačně možnosti daného implementačního prostředí stoprocentně.

Rozpor mezi portabilitou a funkcionalitou lze řešit v použití či nepoužití toho či onoho nástroje dle ostatních okolností, zejména dle zaměření daného projektu.

Buduje-li se rozsáhlý informační systém "na klíč" bez ambicí na jeho další nasazení a požaduje-li se optimální využití všech funkcí a možností zvoleného implementačního prostředí, je vhodné použít APS. V případě tvorby typových pojektů, od nichž se očekává četné nasazení na různých již existujících typech počítačů popř. dořešení určité části informačního systému do již existujícího projektu, není APS nejvhodnějším vývojovým prostředkem.

APS podstatně zvyšuje produktivitu práce v etapě vývoje programového vybavení. Specifikace navrhovaných funkcí je na vysokém stupni formalizace, což přináší obrovské úspory času v etapě údržby programového vybavení. APS dokonale dokumentuje programovou stránku řešení (automaticky!!!).

Výše uvedený projekt byl realizován včetně analýzy kapacitou 3 člověkoroků. Projekt byl implementován do velice progresivního avšak programátorsky složitého prostředí. V případě použití klasického vývojového prostředí (COBOL) by se požadovaná kapacita pro vývoj minimálně zdvojnásobila. Navíc by vznikly veliké kapacitní nároky na následnou údržbu takovéhoto systému.