Ing. Milan Nepovím
České ~~magnetické~~ závody, koncern, CVTES, k.ú.o. Praha
Energoelektronika

ZKUŠENOSTI S APLIKACÍ STRUKTUROVANÉHO PROJEKTOVÁNÍ PŘI TVORBĚ UŽIVATELSKÉHO SW

Každý analytik či programátor ve své práci po jisté době dospěje do stavu, kdy při realizaci zadaného problému se přestane zabývat pouze problémem "CO" řešit, ale začne přemýšlet o tom "JAK" řešit. Přestože v současné době je k dispozici větší počet výkonných prostředků pro zefektivnění práce programátorů (normalizované programování, strukturované programování, modularita, HIPO, postup shora dolů, pomocné knihovny, tým vedoucího programátora apod.), ne každému se podaří uspokojivě vyřešit problém "JAK" svou práci zracionalizovat a dovést ji do takové stadia, že je sám se svým výkonem spokojen. Jedním z důvodů je i to, že zprávy a články, obsahující informace o těchto metodách mají často charakter kusý a informativní, čímž je možnost bezprostředního praktického použití do jisté míry ztížena. Málokdo nalezne odvahu experimentovat na této bázi, když je současně nucen plnit termínované úkoly. Dalším důvodem může být i ta skutečnost, že neexistují kursy či školení, kde by se zájemce seznámil s metodikou a možností aplikace moderní technologie tvorby programových prostředků. Tím je odkázán sám na sebe a jeho kvalitativní růst je otázkou délky praxe. Protože tento fakt je z hlediska nedostatku řešitelských kapacit neúnosný, pokusili jsme se vytvořit metodiku projektování, která by stávající situaci pomohla zlepšit.

V procesu tvorby SW jsme realizovali známé poznatky s moderní technologie analýzy a programování a po několikaletých zkušenostech bych se chtěl zmínit o našich výsledcích, které nám pomohly efektivně vyřešit programový celek SPEIN (Systém plánových a ekonomických informací). Tyto poznatky jsou neustále dopracovávány do cílové podoby a budou sloužit všem analytikům a programátorům ČEZ jako metodické materiály. Nová metodika je platná pro navrhování subsystémů, řešených dávkovým způsobem zpracování.

Při návrhu nového stylu práce pro tvorbu programového vybavení jsme dospěli k názoru, že musíme realizovat následující opatření :

1. strukturalizovat proces tvorby SW do relativně izolovaných fází
2. podpořit jednotlivé fáze
   - souborem pravidel a zásad, podle kterých by činnosti byly v dané fázi realizovány
   - definováním všech vstupů a výstupů k dané fázi
   - formalizací způsobu evidence a dokumentování průběhu a výsledku činností v dané fázi.

Výsledkem našeho úsilí byl návrh způsobu práce, který jsme nazvali "Strukturované projektování".

Pojem Strukturované projektování není dosud běžným termínem v naší literatuře, která pojednává o tvorbě SW. Je to zásadní přístup k řešení a k tvorbě SW. Termín "Strukturované projektování" byl použit záměrně, aby se odlišil od často citovaného pojmu "Strukturované programování". Důvodem je, že mnoho pracovníků používá tento termín k označení rozdílných skutečností. Ve většině případů je "Strukturované programování" chápáno jako styl práce, související pouze s tvorbou programu, tj. s návrhem algoritmu a kódováním.

Strukturované projektování zahrnuje v sobě řadu samostatných podpůrných metod. Tyto metody lze využít jak všechny současně pro celý postup řešení SW, tak i izolovaně pro určitou fázi procesu tvorby. Jsou natolik samostatné, že mohou fungovat bez ostatních, za podmínek vyhovujícího okolí, např. předepsané vstupy.

Na tomto místě bych rád zdůraznil následující skutečnost. Mnozí projektanti se domnívají, že použitím jedné moderní metody udělali vše pro zvýšení své produktivity a povýšili svou práci na úroveň formalizovaného inženýrského postupu. Pravdou zůstává, že svou činnost do jisté míry mohli zracionalizovat, ale může se však stát, že v dalších etapách své práce narazí na problémy, které by vůbec nenastaly, kdyby do předchozí činnosti nezapojili "racionalizační techniku". Je nutné domyslet dopad použití racionalizační metody na celý technologický proces tvorby SW a na cíle, kladené na výsledný SW produkt.

Soubor metod, tvořících strukturované projektování (viz obr. 1) je rozčleněn do jedenácti oblastí, které lze shrnout do dvou skupin podle způsobu, jakým se podílí na procesu tvorby SW. První skupinu tvoří metody, související s technickou stránkou procesu návrhu, vývoje a testování, druhá skupina se vztahuje k organizační a řídící stránce tohoto procesu :

1. metody související s technickou stránkou tvorby SW
   - analýza
   - strukturovaný návrh
   - strukturované programování
   - testování
2. metody podpůrné, související s organizační a řídící technikou tvorby SW
   - týmová práce
   - kontrolní činnost

- konečná dokumentace
- pracovní dokumentace
- podpůrné knihovny
- HIPO
- řízení procesu tvorby SW.

Řízení projektu. Základním předpokladem racionální práce je vyřešení problematiky zahrnuté do oblasti Řízení projektu, kterou jsme rozdělili do dvou částí a to činností souvisejících s vlastním řízením a činností souvisejících s technologií SW. V současné době platné předpisy strukturalizují proces projektování do etap, které jsou příliš hrubé a náročné na řešitelská kapacity. Tím i kontrola je iluzorní, neboť efekt kontroly provedené např. až po spotřebování 2000 hodin je nejistý, neboť málokdo odmítne navrhované řešení po spotřebování takového počtu hodin. Proto jsme provedli strukturalizaci celého procesu do etap kapacitně méně náročných a problémově ohraničených na menší úseky (viz obr. 2).

Celý proces začíná zpracováním požadavků, kdy řešitel posuzuje formalizované požadavky a vytváří první návrh podle představ uživatele ve formě hierarchického schematu. V další etapě se navrhuje jedno nebo více řešení a to vždy se závěrem, že za daných podmínek a omezení

1. je možné v projektu dále pokračovat
2. je nutné další práci zastavit
3. je nutné opakovat tuto etapu, neboť výsledky jsou nedostatečné.

Po odsouhlasení výsledků je pokračováno další etapou, v níž je vytvořen návrh celého subsystému, ze kterého vyplynou požadavky na organizační, technické a programové zabezpečení. Realizačním výstupem je v podstatě technický projekt.

Další etapa obsahuje činnosti související s přípravou programového zabezpečení. Je dopracováno hierarchické schema programového systému, doplněné HIPO diagramy včetně navrženého interface. To vše slouží pak jako podklad pro vypracování specifikací jednotlivých částí pro programování. Poslední dvě fáze, tj. programování a testování jsou standardní činnosti, podpořené v systému strukturovaného projektování formalizovanými postupy.

V poslední etapě jsou zajištěny veškeré nutné podmínky pro uvedení subsystému do provozu, tj. zajištění organizačních, statutárních i právních úprav, zajištění HW, lidí i financí i realizace školení pracovníků výpočetního střediska a uživatele. Konečná fáze je zkušební provoz a později pak uvedení do provozu rutinního.

Vlastní řízení je v současné době realizováno pouze ve fázi plánování a v důsledné kontrole a není dosud podpořeno metodickými materiály. Zůstává tedy oblastí značně ovlivněnou osobností vedoucího.

Strukturalizací procesu tvorby SW a vložením etapy studie proveditelnosti jsme dosáhli snížení ztráty kapacit, přesnějšího plánování kapacit i konečného termínu, možnosti přesunu kapacit v případě ohrožení termínu a efektivnější kontroly (chyba je nalezena co nejblíže místa vzniku).

Vzhledem k omezení rozsahu příspěvku není možné všechny části strukturovaného projektování popisovat, zmíním se proto pouze o některých a to formou krátkého vysvětlení metody se stručným uvedením vlastních zkušeností.

Strukturovaný návrh. Návrhem programového systému je chápán proces transformace a ověřování, kdy jsou požadavky uživatele převáděny do tvaru, ve kterém by měly být realizovány pomocí počítače. Strukturovaný návrh programového systému je formalizovaný postup, který redukuje počet

intuitivních rozhodování v návrhu. Je to souhrn úvah a technik obecné tvorby programů pro jednodušší a rychlejší návrh, kódování, testování a modifikaci. Členění složitého celku do menších, snadno zvládnutelných částí je naprosto nutné. V současné době je však dělení do subčástí realizováno podle volné úvahy autora návrhu. Neměl by to být problém a řešení je ve většině případů úspěšné. Mohou se však objevit problémy v následujících etapách při řešení nižších úrovní, neboť pracovníci, navazující na prvotní práci autora, by museli vycházet ze stejné úvahy. Proto je snaha formalizovat proces dekompozice a návrhu programového systému, aby všichni, kteří se podílejí na návrhu, programování, testování a údržbě, zaujímali stejný postoj a realizovali stejnou technologii při návrhu programového systému.
Tuto oblast metodiky jsme zčásti upravili v duchu nových názorů a myšlenek z loňského semináře Programování 80.

Při členění složitého celku jsme narazili na skutečnost, že ne každý je ihned schopen realizovat dekompozici systému i když podle pravidel a kritérií kladených na výsledek. Domníváme se, že to nesouvisí pouze s konservatismem lidí, ale spíše s tím, že tento způsob řešení je nový a zásadní a každý ho musí dostatečně si osvojit a vyzkoušet.

Dělením do menších částí byla umožněna výrazná dělba práce, paralelní testování různých částí, efektní využití ladících knihoven, dosažení přehledné dokumentace, důkladné testování.

Strukturované programování. Strukturovaným programováním rozumíme postoj k vytváření algoritmu a postoj ke kódování s ohledem na komunikaci s lidmi, nikoliv s počítačem. Princip strukturovaného programování ve fázi návrhu algoritmu je možné shrnout do několika zásad :

1. Obecné zásady vytyčují požadavky kladené na správně konstruovaný algoritmus (determinovanost, obecnost, jednoznačnost, konečnost).

2. Princip vývoje "shora dolů" spočívá v tom, že musí být souvislost mezi algoritmem a tokem řízení, které by se měly odvíjet směrem shora dolů.

3. Princip jednoho vstupu a jednoho výstupu je zdůraznění faktu, že komunikace subčástí musí být realizována pouze přes jednoznačně definovaný interface.

4. Princip jednoduchosti odpovídá požadavku na nejvyšší srozumitelnost algoritmů a kódů.

5. Omezení velikosti je opět princip poplatný požadavku srozumitelnosti a přehlednosti z pohledu člověka, nikoliv počítače.

6. Omezení skoků je zásada, odvozená z principu "postup shora dolů" a znamená co nejvíce omezit skoky

7. Princip řídících struktur znamená, že většinu algoritmů lze vyjádřit pěti tzv. řídícími strukturami, které definují pořadí provádění příkazů. Jedná se o sekvenci, rozhodování, řízení smyček DOWHILE, řízení smyček DOUNTIL, rozhodování s vícenásobným větvením.

8. Zajištění spolehlivosti vychází z toho, že výsledek nebude bez chyb. Proto výsledný návrh musí být konstruován tak, aby mohl pracovat s předvídatelným výsledkem i při výskytu chyb. Je proto nutno dodržovat principy detekce a izolace chyb.

V oblasti kódování jsou stanovena pravidla pro formální úpravu programů, (jako např. komentáře, opis specifikace, ladící výpisy, návratové kódy...) a je stanoven postup pro převod algoritmu do kódu určeného jazyka a předepsány jazykové prostředky, kterými jsou řídící struktury transformovány

do příslušného jazyka.

Při posuzování konečného efektu z použití strukturovaného programování zjistí se příznivý vliv ve třech oblastech :

1. vliv na výsledný SW se projeví v lepší kvalitě, větší spolehlivosti, snadné a rychlé údržbě, rychlejší implementaci a v nižších nákladech řešitele na vývoj;

2. vliv na programátory znamená vyšší produktivitu, lepší morálku, efektivnější testování a přesnější a užitečnější dokumentaci;

3. vliv na uživatele - je rychlejší a pružnější odezva na nové uživatelovy požadavky a nižší uživatelovy náklady na vývoj a údržbu.

Testování. V této závěrečné činnosti se plně projeví kladný vliv formalizace předchozích kroků (strukturovaný návrh, strukturované programování) a pomocného aparátu knihoven, pracovní dokumentace a týmové práce. Bez nich není možné úspěšně zvládnout testování, tzn., že nelze řešit testování jako izolovaný problém.

Při vlastním testování jsme důsledně aplikovali metodu top-down, jejíž princip je následující :

Všechny moduly, které tvoří strukturu programu se transformují a to ve tvaru, kdy obsahují pouze tisk informace o tom, že jim bylo předáno řízení. V tomto redukovaném tvaru se uloží do knihovny ve tvaru "load". Hlavní modul, obsahující všechny funkce se dodá ze vstupního proudu a pomocí katalogizované procedury je v prvním kroku přeložen. V dalším kroku jsou k němu pomocí linkege editoru přisestaveny z "load" knihovny všechny zbylé moduly a program je spuštěn na testovacích datech. Po otestování hlavního modulu je tento uložen do load knihovny a postup je opakován pro všechny ostatní moduly.

Kontrolní činnost. Mechanismus kontrol je formalizován do etap: plánování, příprava, vlastní kontrola, rozhodnutí o dalším postupu, oprava, pokračování.

Důsledně jsme realizovali kontroly uvnitř pracovního týmu. Cennou zkušeností je, že se vyplatí oznámit autorovi předem kriteria, podle kterých pak bude posuzován výsledný produkt. Vypadá to jako zdůraznění něčeho samozřejmého, ale zkusme se zamyslet, kolikrát dostane řešitel do ruky zadání, kde jsou vyjmenovány pouze funkce, které musí naprogramovat, ale už tam není nic o nárocích na paměť, CPU, periferie, formální úpravu programu atd.

Během několikaletých zkušeností jsme poznali, že kontroly mají příznivý vliv na členy týmu v tom, že jejich odborný růst je rychlejší. Jejich názory na řešení jsou totiž neustále konfrontovány s názory druhých a tím se seznamují se stále novými možnostmi řešení.

Nelze také pominout význam kontrol, který mají pro spolehlivost výsledného produktu z pohledu programovacího stylu "egoless". Vlivem kontrolní činnosti autor už se necítí pouze jediným autorem, kritika jeho díla se mu nejeví jako kritika jeho osobnosti a tímto nastoupená cesta důvěry a otevřené diskuse nad řešením zefektivní produktivitu pracovního týmu.

O významu kontrol pro řízení není snad nutné se zvláště zmiňovat. Chtěl bych ale znovu pozornit na to, že kontrola, jež se provádí ihned po realizaci dané etapy má za následek, že chyba se zjistí co nejblíže místa vzniku a není nutné ji pracně hledat prostřednictvím vedlejších účinků vyvolaných v následujících etapách.

Pracovní dokumentace je souhrn pracovních záznamů, poznatků, zkušeností a výsledků, vytvářených během určité

vymezené etapy pracovního procesu. Vzniká hned na počátku při zahájení prací v dané etapě procesu tvorby SW a je průběžně doplňována s postupem rozpracovanosti. Účelem vedení pracovní dokumentace je zajistit, aby každý pracovní dokument, který se vytvoří v jednotlivých fázích tvorby SW bylo možno snadno a jednoznačně identifikovat. Vytváří se v tolika hierarchických úrovních, kolik je jich vyžadováno při projektování metodou shora dolů. Pro práci s pracovní dokumentací není nutná podobná dokumentace v předchozím stupni.

Formalizace pravidel pro vedení pracovní dokumentace pomáhá vychovávat k systematické práci, která je v naší profesi nutná. V podstatě nutní autora k tomu, aby se každým dokumentem zabýval podrobněji a dokázal ho zařadit tak, aby pomocí odkazů ho bezpečně našel.

Existence pracovní dokumentace umožní :

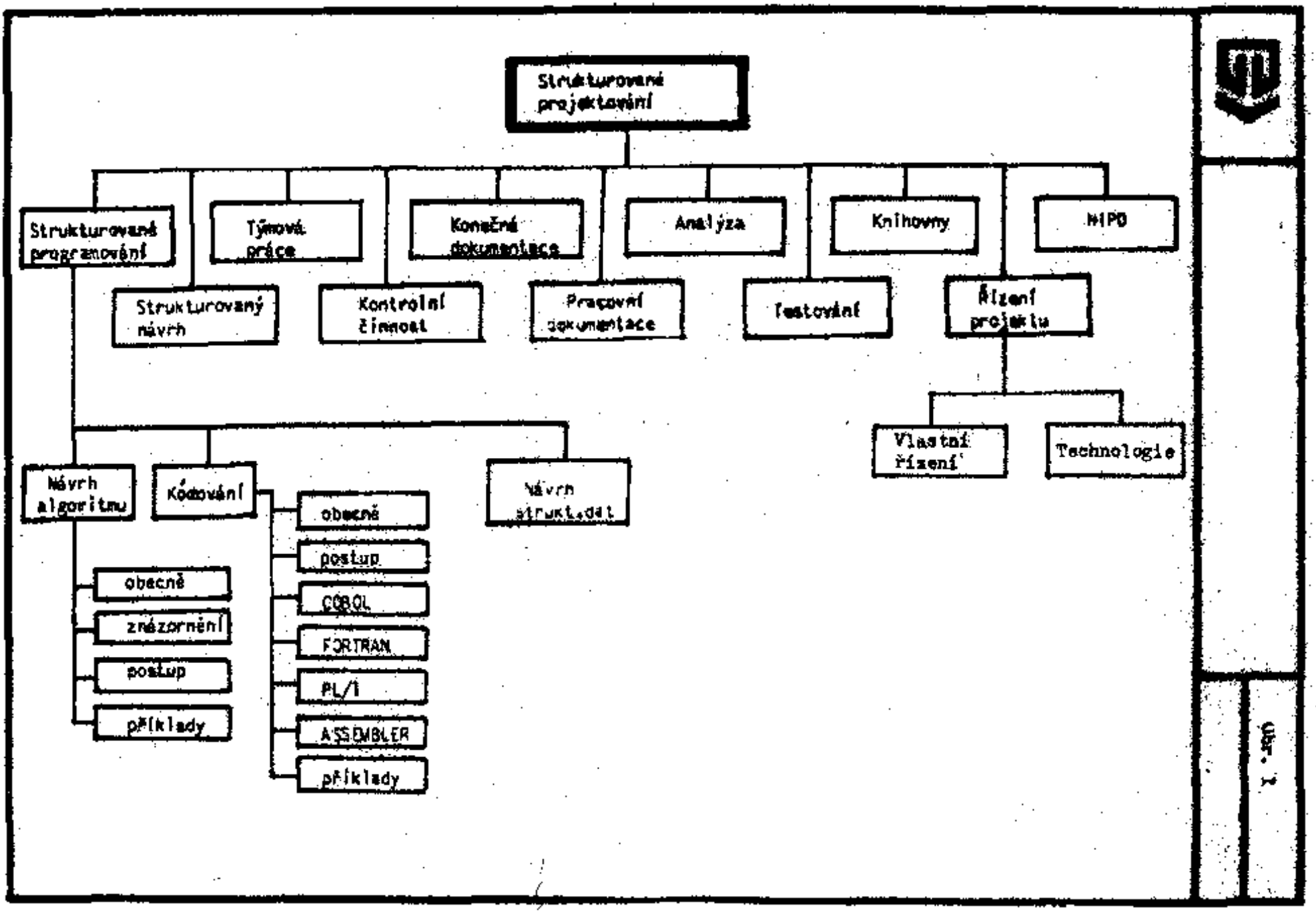
pokračovat v práci i někomu jinému než je autor;
dostatečnou kontrolu o stavu prací na projektu i kontrolu tempa, kterým je práce vedena;
snadné zásahy a opravy stávajících částí;
rychlé vytvoření konečné dokumentace;
efektivnější práci projektanta (formalizovaný postup a předepsané formuláře).

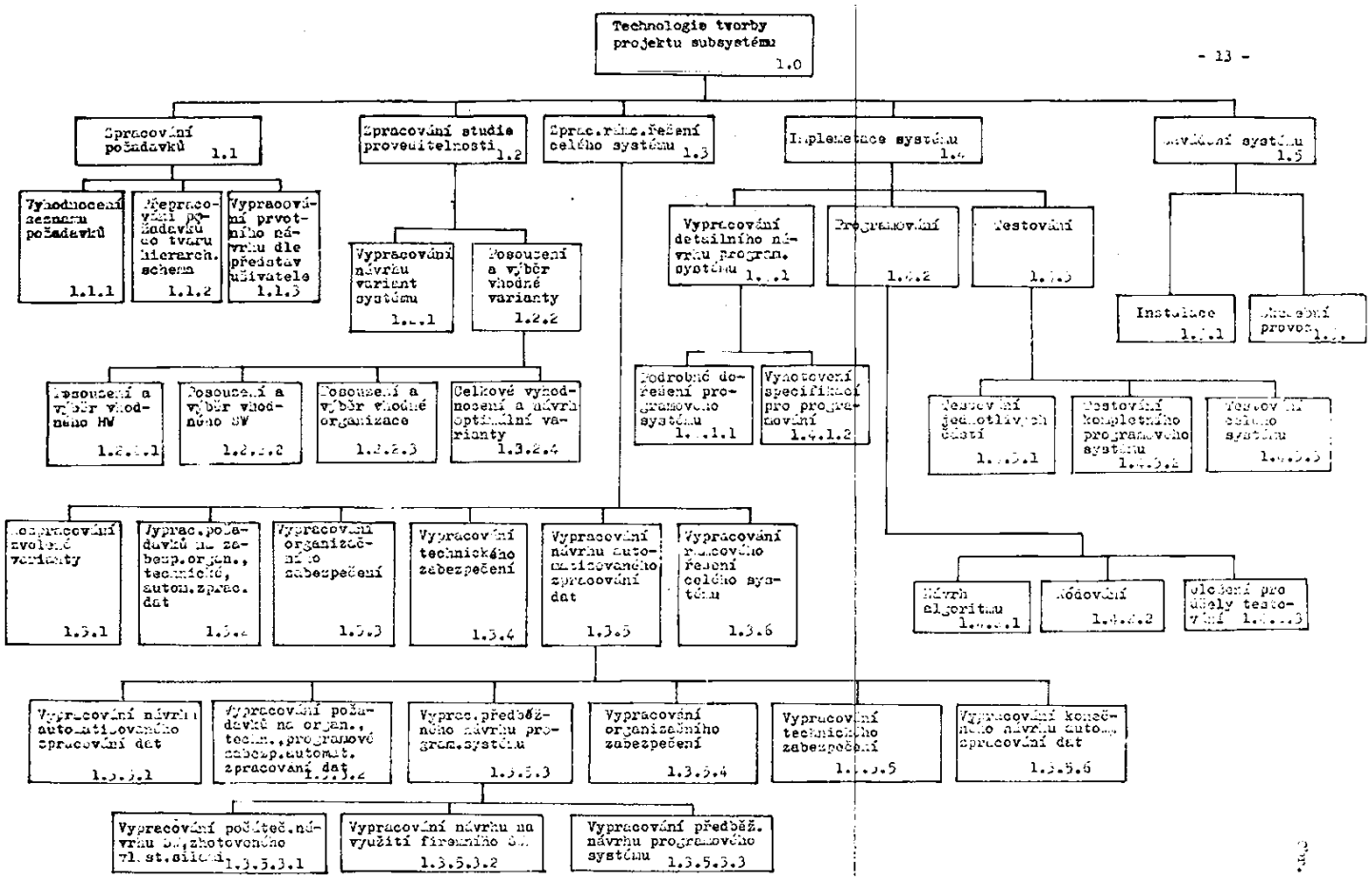
Knihovny. Systém podpůrných knihoven obsahuje jednak návody (procedury), kterými je řízena určitá činnost a dále archivované výsledky těchto činností. Knihovny procedur obsahují počítačové procedury (aktualizace složek vnitřní knihovny, výpisy stavu knihoven, testovací procedury) a procedury administrativní (pokyny pro práci s vnějšími knihovnami, zakládání výpisů, postupy při testování). Knihovny výsledků, obsahující aktuální stav vyvíjeného SW se dělí na část

interní (SW uložený ve tvaru vhodném pro počítač) a na část externí, která je odrazem stavu vnitřní knihovny ve formě vhodné pro projektanta. Za hlavní přínos je možné považovat :

1. sjednocení způsobu práce a tím získávání návyků, vyplývajících z opakovatelnosti prací a tím zvýšené produktivity, neboť nenechávají prostor pro neracionální přístup k řešení,
2. zlepšení řídící činnosti, neboť je možno kontrolovat stav výsledků i dodržování technologické kázně, která vyplývá z postupů, uchovávaných v knihovnách;
3. poskytování informací o aktuálním stavu vytvářených výsledků. Tvořeným programovým systémem probíhá nepřetržitá integrace, stav systému odpovídá stavu podpůrné knihovny.

Závěrem bych chtěl zdůraznit, že všechny metody bez rozdílu mají nesporný vliv na zvyšování kvality projektování. Je to důsledek sjednoceného a formalizovaného způsobu práce a zlepšené řídící činnosti. Způsob naší práce nedosáhl ještě cílového stavu, některé oblasti musíme podpořit dokonalejšími písemnými materiály, přesto se však domnívám, že značně přispěl k tomu, že projektování se stalo více profesionální záležitostí než neprocedurální posloupností činností uměleckého charakteru.


Strukturované projektování
Strukturované programování
Týmová práce
Konečná dokumentace
Analýza
Knihovny
HIPO
Strukturovaný návrh
Kontrolní činnost
Pracovní dokumentace
Testování
Řízení projektu
Vlastní řízení
Technologie
Návrh algoritmu
Kódování
Návrh struktur dat
obecně
postup
COBOL
FORTRAN
PL/1
ASSEMBLER
příklady
obecně
znázornění
postup
příklady


Technologie tvorby projektu subsystému 1.0
Zpracování požadavků 1.1
Zpracování studie proveditelnosti 1.2
Zprac.rámc.řešení celého systému 1.3
Implementace systému 1.4
Zavádění systému 1.5
Vyhodnocení seznamu požadavků 1.1.1
Přepracování požadavků do tvaru hierarch. schema 1.1.2
Vypracování prvotního návrhu dle představ uživatele 1.1.3
Vypracování návrhů variant systému 1.2.1
Posouzení a výběr vhodné varianty 1.2.2
Vypracování detailního návrhu program. systému 1.4.1
Programování 1.4.2
Testování 1.4.3
Instalace 1.5.1
Posouzení a výběr vhodného HW 1.3.2.1
Posouzení a výběr vhodného SW 1.3.2.2
Posouzení a výběr vhodné organizace 1.3.2.3
Celkové vyhodnocení a návrh optimální varianty 1.3.2.4
Podrobné dořešení programového systému 1.4.1.1
Vypracování specifikací pro programování 1.4.1.2
Testování jednotlivých částí 1.4.3.1
Testování kompletního programového systému 1.4.3.2
Testování celého systému 1.4.3.3
Vypracování zvolené varianty 1.3.1
Vyprac.požadavků na zabezp.orgán., technické, autom.zprac. dat 1.3.2
Vypracování organizačního zabezpečení 1.3.3
Vypracování technického zabezpečení 1.3.4
Vypracování návrhu automatizovaného zpracování dat 1.3.5
Vypracování průmyslového řešení celého systému 1.3.6
Návrh algoritmu 1.4.2.1
Kódování 1.4.2.2
Uložení pro účely testování 1.4.2.3
Vypracování návrhu automatizovaného zpracování dat 1.3.5.1
Vypracování požadavků na orgán., techn., programové zabezp.automat. zpracování dat 1.3.5.2
Vypracování předběžného návrhu program.systému 1.3.5.3
Vypracování organizačního zabezpečení 1.3.5.4
Vypracování technického zabezpečení 1.3.5.5
Vypracování konečného návrhu automat. zpracování dat 1.3.5.6
Vypracování počítač.návrhu na zhotovovacím zař.st.sítě 1.3.5.3.1
Vypracování návrhu na využití firemního SW 1.3.5.3.2
Vypracování předběž. návrhu programového systému 1.3.5.3.3