2. DŮVODY ZMĚN V ANALÝZE SYSTÉMŮ A PŘECHOD KE STRUKTUROVANÝM TECHNIKÁM

Cílem vývoje informačních systémů je vytvořit nový systém, který bude fungovat lépe a radostněji než starý. Projektant využívá abstrakce a rozkladu celku na části, analyzuje problém a syntetizuje řešení. Konkrétní metodická podoba těchto věčných principů závisí na tom, jaké kulturní dědictví a jaké nástroje má projektant k dispozici.

Metody, techniky a prostředky používané pro analýzu a vývoj informačních systémů prošly v posledních dvou desetiletích značným vývojem. A předpokládá se, že vývoj systémové analýzy bude značný hlavně ve druhé polovině 90. let.

Do konce 70. let začínala většina projektů vývoje systému slovním popisem uživatelských požadavků. Tedy analytici popisovali to, jak pochopili systém i jak má nový systém vypadat, souvislým textem. Proto byly vytvořeny (ve 2. polovině 70. let) první strukturované přístupy (De Marco, Gane a Searson), které chtěly odstranit problémy stávajících přístupů:

- Jednolitost: k pochopení problému bylo nutné číst funkční specifikaci celou od první až do poslední strany. Ale v řadě případů analytik nebo uživatel potřebují přečíst jen určitou část specifikace bez vztahu k jiným částem. *)
- Redundance: Stejné informace se často opakují v různých částech dokumentace. Důsledkem je, že při každé změně uživatelských požadavků

*) To ovšem předpokládá, že jsou schopni z dokumentace rozpoznat, která část je pro daný problém ta pravá.

(z jakéhokoliv důvodu) je třeba měnit několik různých částí dokumentace (když někdo ví, kde všude se daná věc vyskytuje). Jelikož je velice těžké takové změny provádět skutečně precizně, vede tento problém často k nekonzistenci dokumentace.

- Nejednoznačnost: podrobné uživatelské požadavky byly často jinak interpretovány uživatelem, analytikem, designérem (návrhářem) programového systému a programátorem. Podle výzkumů na konci 70. let bylo víc jak 50% chyb v OS a 75% změn v nákladech způsobeno špatným pochopením funkční specifikace nebo chybami ve funkční specifikaci.
- Nemožná údržba: Na základě všech výše uvedených důvodů byla funkční specifikace zastaralá již na konci procesu vývoje systému (tj. při zavedení systému do používání) a často již na konci fáze analýzy systému. Většina systémů vyvinutých v 60. a 70. letech nemá žádné údaje o cílích organizace, které podporuje. A protože původní analytici a uživatelé již dávno zmizeli a dokumentace neexistuje, provádí se údržba formou chaotických zásahů.

V době, kdy se o těchto problémech začalo diskutovat, se začaly zavádět nové myšlenky do oblasti programování a návrhu programových systémů. Tyto myšlenky vyústily do strukturovaného programování a strukturovaného designu a slibovaly značné zlepšení organizace, kódování, testování a údržby počítačových programů. Strukturované programování a design přinesly úspěch, ale řada organizací automatizovaného zpracování dat rychle dospěla k názoru, že nemá smysl psát skvělé programy a navrhovat vysoce modulární systémy, jestliže nikdo skutečně neví, co měly požadované systémy dělat. Strukturované a modulární programování vytvoří dobře udržovatelnou strukturu pro danou úlohu. Vlastním vymezením úlohy se však nezabývá. Mezi cíli, funkcemi, uživatelským rozhraním projektu a strukturovaně programovaným řešením je propast, překlenutá vývojem metod v 70. letech.

Výsledkem byl postupný přechod (trval asi 10 let) k funkční specifikaci (funkčnímu návrhu). Funkční specifikace se snaží být:

- grafická: složená z různých diagramů, doplněná podrobnými textovými materiály, které v mnoha případech slouží spíše jako reference, než jako hlavní část specifikace;
- postupná: samostatné části specifikace lze číst nezávisle na ostatních částech;
- minimálně redundantní: každá změna v uživatelských požadavcích se dotkne pouze jedné části specifikace.

Tento přístup - "strukturovaný návrh systému" je dnes používán ve většině organizací vytvářejících informační systémy.

Ve strukturovaných přístupech 70. a začátku 80. let zůstávala ještě nevyřešena řada problémů:

- přístupy zdůrazňovaly tvorbu modelu stávajícího stavu. Tento model je ještě velmi důležitý pro pochopení fungování systému. Avšak jeho důsledná tvorba skrývá velké nebezpečí: - často trvá velice dlouho, než uživatelé vidí výsledky práce, a ukončí projekt ještě dříve, než se může přejít ke studiu a návrhu nového stavu ("na co studujete starý uživatelský systém, když stejně každý ví, že bude nahrazen novým"). Proto je třeba zkrátit studium současného stavu na minimum a spojit jej rovnou s analýzou požadovaného stavu;
- byl vágně definován rozdíl mezi logickým a fyzickým modelem. (Logický model je nezávislý na implementační technologii. Fyzický model je založen na předpokladech daných implementačním prostředím a technologií implementace.) V polovině 80. let se vžily termíny podstatné (esenciální) modely (Essential models) místo logických a implementační modely místo fyzických. (Esenciální model vyjadřuje podstatu - esenci - systému);
- "klasické" strukturované přístupy neobsahují techniky vývoje systému pro tvorbu real-time systémů.Nemají prostředky pro vyjádření přerušení a signálů a pro vyjádření synchronizace a koordinace různých procesů. Yourdonova strukturovaná metoda, kterou zde budeme popisovat jako reprezentantku moderních strukturovaných přístupů, zavádí pro real-time systémy novou notaci a nové prostředky modelování - řídící toky, řídící procesy a diagramy přechodu stavů;
- "klasické" strukturované přístupy zahrnovaly většinou pouze modelování funkcí systému. Datové modelování bylo primitivní a často bylo potlačováno nebo úplně vynecháno. A zatím víc a víc organizací potřebovalo, aby jejich systémy zahrnovaly komplexní funkce a komplexní vztahy mezi daty a komplexní real-time charakteristiky. Proto byly do strukturovaných přístupů přidány diagramy entit a vztahů (ERD) a STD diagramy a tyto prostředky byly integrovány tak, aby se vzájemně doplňovaly;
- v klasických přístupech byl postup stejný jako postup prezentace návrhu - striktně shora dolů (kontextový diagram, úroveň 0 atd.). Tento postup ale ne vždy funguje (zvlášť u složitých, tvůrcům málo známých systémů). Proto vznikl nový přístup k návrhu - rozdělování systému dle událostí.

Strukturované přístupy jsou většinou dost "nepříjemné" z hlediska pracnosti návrhu (hlavně pokud se týká změn v obrázcích). Velice podstatné ulehčení práce a zvýšení její účinnosti při vývoji systému přinesly prostředky typu CASE (Computer Aided Software Engineering). Prostředky typu CASE unožnují lehké promítání změn, automatické kontroly konzistence atd. Řada z nich obsahuje i prostředky pro generování zdrojových programů přímo ze specifikace, což snižuje dobu a náklady potřebné na programování.

Jak již bylo uvedeno, v 70. letech se vyvíjely jednak strukturované přístupy k analýze systému a jednak strukturované programování a design. Hlavním problémem bylo, že strukturovaná analýza se zabývala návrhem velkých, komplexních systémů, zatímco programování a design se víc týkaly jednotlivých programových systémů pracujících na jednom procesoru. Chyběl most mezi analýzou informačního systému a návrhem programu - system design. Dnes již existuje několik prací, které tento problém řeší (Page-Jones 1988, Yourdon a Constantine 1989).