1 Úvod do softvérového inžinierstva

Čo je to softvérové inžinierstvo a softvér?

☞ *Systematický prístup k vývoju, prevádzke, údržbe a vyradeniu softvéru* (IEEE, 1994).

Aplikovanie vedy a matematiky, pri ktorom sa prostredníctvom programov, postupov a s nimi združenej dokumentácie možnosti počítačového vybavenia stávajú užitočnými človeku (Boehm, 1981)

Softvérové inžinierstvo a softvérová kríza

Softvérové inžinierstvo sa nezaoberá iba tvorbou softvérových systémov, ale zahŕňa aj tvorbu softvéru efektívnym spôsobom. Keby sme mali neohraničené zdroje, väčšina problémov s tvorbou softvéru by sa dala vyriešiť. V skutočnosti však treba vytvoriť softvér podľa požiadaviek s ohraničenými zdrojmi a určeným časom ukončenia.

Softvérové inžinierstvo sa zaoberá tvorbou väčších softvérových systémov čo najefektívnejším spôsobom. Základným cieľom je

- zlepšenie akosti softvéru, najmä čo sa týka spoľahlivosti a
- zvýšenie produktivity softvérových inžinierov.

Rozširovanie oblastí použitia počítačov viedlo k riešeniu čoraz zložitejších úloh, pričom návrh a vývoj programov sa zakladal v podstate na ad hoc technikách. V procese tvorby softvéru bolo málo štandardizácie a teda s každým programom sa vymýšľalo už vymyslené. Dôsledkom je zlyhanie celého radu softvérových projektov, predĺženie mnohých z nich. Aj tie softvérové systémy, ktoré sa vytvorili, boli často nespoľahlivé, vyskytovali sa mnohé nepredvídané poruchy, boli slabo dokumentované, nespĺňali požiadavky, ktoré sa na ne kládli.

Na riešenie uvedených problémov bolo treba prehodnotiť všetky oblasti tvorby softvéru. Dôsledkom je vznik softvérového inžinierstva v súvislosti s diskusiami o tzv. softvérovej kríze (konferencie NATO v rokoch 1968-1969), keď sa prvýkrát otvorene poukázalo na metódy spojené s tvorbou programov.

Softvérové inžinierstvo a ostatné inžinierske disciplíny:

nemožnosť uvažovania všetkých podmienok, alternatív

cieľom je minimalizácia škody pri neočakávaných podmienkach

Softvérové výrobky

- **generické:** softvér sa predáva ľubovoľnému záujemcovi
- **zákaznícke (na objednávku):** softvér sa vytvára na základe požiadaviek pre konkrétneho zákazníka.

Softvér

Zbierka počítačových programov, procedúr, pravidiel a s nimi spojenou dokumentáciou a údajmi (IEEE, 1994).

Softvér zahŕňa napr.: požiadavky, špecifikácie, opisy návrhu, zdrojový text, testovacie údaje, príručky, …

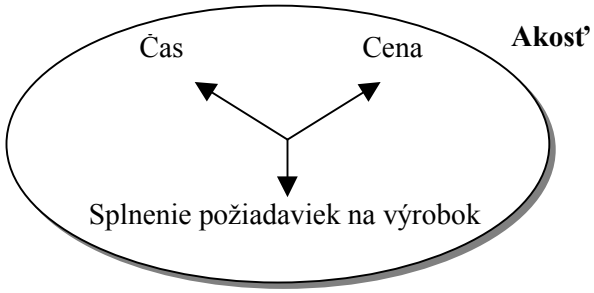
Do 80-tych rokov sa väčšina softvérových systémov vytvárala na objednávku. S rozšírením osobných počítačov sa rozšírili generické softvérové produkty. Problémy spojené s tvorbou softvéru sú podobné pre generické softvérové systémy aj softvér na objednávku. Najväčším rozdielom je, že špecifikácia generického softvéru sa vytvára iba v rámci spoločnosti, ktorá produkuje softvér.

Generické produkty odrážajú požiadavky na to, čo by chceli predávať. Musia byť navrhnuté pružne, aby sa dali zakomponovať zmeny. Na druhej strane špecifikácia systému na objednávku je často súčasťou zmluvy (kontraktu) so zákazníkom. Definuje sa podrobne a každá zmena sa podrobne vyhodnocuje (spolu s odhadom nákladov).

Akosť

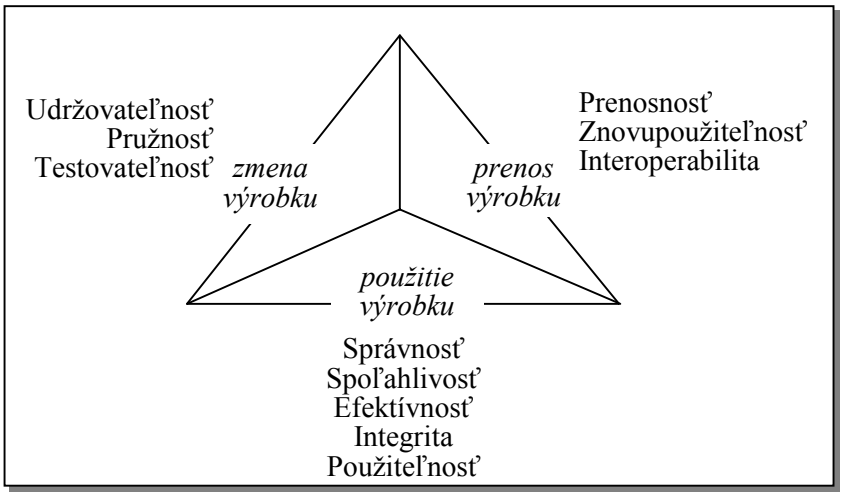
Akosť je súhrn vlastností a charakteristík výrobku, procesu alebo služby, ktoré preukazujú jeho schopnosť splniť určené alebo odvodené potreby (ISO 8402).

- Akosť nie je definovaná ako absolútna miera, ale ako stupeň splnenia požiadaviek, resp. potrieb.
- Ak cieľom bude vyvinúť nefunkčný softvér, potom čím menej bude fungovať, tým je kvalitnejší.



Vlastnosti softvéru

Vlastnosti softvéru predstavujú tie charakteristiky, ktoré sa vyžadujú akonáhle sa softvér začne používať. Nejde o služby, ktoré softvér zabezpečuje.



Správnosť': miera, do akej výrobok spĺňa špecifikáciu.

Spoľahlivosť': správanie sa výrobku pri výpadku – výrobok by nemal pri výpadku systému spôsobiť ani fyzické ani ekonomické škody.

Integrita: úsilie, ktoré treba vynaložiť na to, aby systém bol bezpečný.

Efektívnosť': splnenie kritérií na využitie zdrojov počítačového systému, na čas potrebný na realizáciu a ďalších kritérií spojených so samotným vývojom výrobku (napr. náklady).

Použiteľnosť': úsilie, ktoré treba vynaložiť na to, aby sa dal výrobok používať.

Prenosnosť': úsilie, ktoré treba na prenos výrobku z jednej platformy na inú (prostredie, v ktorom sa prevádzkuje).

Znovupoužiteľnosť': miera, do akej možno jednotlivé časti výrobku znovu použiť iných podobných aplikáciách.

Interoperabilita: úsilie, ktoré treba na zabezpečenie spolupráce systému s inými systémami.

Udržovateľnosť': úsilie, ktoré treba vynaložiť na ďalší vývoj a údržbu výrobku podľa meniacich sa potrieb zákazníka a aj meniaceho sa okolia.

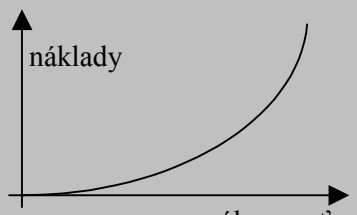
Pružnosť': úsilie, ktoré treba na modifikáciu výrobku v prevádzke (napr. zvýšenie jeho funkcionality).

Testovateľnosť': úsilie, ktoré treba vynaložiť na testovanie vlastností výrobku, napr. či vykazuje požadované správanie.

Dôležitá je aj **dokumentovateľnosť softvéru**, t.j. miera, do akej sú všetky rozhodnutia počas vývoja zdokumentované a kontinuita dokumentácie počas všetkých etáp.

Optimalizácia všetkých vlastností sa sťažuje tým, že niektoré sa navzájom vylučujú. Napr. keď vytvoríme lepšie používateľské rozhranie pravdepodobne sa zníži výkonnosť. Podobne je to napr. s nákladmi na vývoj softvéru a výkonnosťou softvéru. Náklady zlepšovaním výkonnosti rastú exponenciálne.

Vzťah medzi nákladmi na vývoj softvéru a výkonnosťou softvéru



Program
(používa ho autor, v podmienkach, pre ktoré ho vyvinul)

Programový výrobok
(program, ktorý môže používať, opravovať, rozširovať hocikto; lepšie otestovaný, dokumentácia)
3 krát drahší ako program

Programový systém
(zbierka spolupracujúcich programov, ktoré sú koordinované definovaným spôsobom; rozhrania musia zodpovedať dohode; tiež zdroje, ktoré využíva musia zodpovedať dohodnutým požiadavkám)
3 krát drahší ako program

Programový systém - výrobok
9 krát drahší ako program

Problémy s tvorbou softvéru

Problémy spojené s konštruovaním veľkých softvérových systémov vyvstávajú najmä z ich vnútornej vlastnosti, ktorou je zložitosť. Je nemožné, aby jeden človek udržiaval všetky podrobnosti o každom aspekte takéhoto projektu vo svojej hlave. Navyše tvorca softvérového systému by mal mať znalosti z viacerých oblastí ako napr. výpočtová technika, riadenie, komunikácia, všeobecné metódy riešenia problémov, navrhovanie. Dôsledkom je, že jeden človek to nemôže zvládnuť a teda tvorba programov nemôže byť záležitosťou jedného programátora. Treba využívať znalosti tímu odborníkov. A tak vznikajú problémy s organizáciou viacerých ľudí pri práci v tímoch.

Podstatné, vnútorné problémy – pravdepodobne nevieme odstrániť (Brooks, 1975):

- **zložitosť:** žiadne dve časti nie sú rovnaké, zložitosť je zdrojom ďalších problémov ako napr. komunikácia v tímoch, problém porozumenia všetkých možných stavov systému, problémy s rozširovaním, atď.
- **prispôsobivosť:** ak sa niečo zmení, často sa musí prispôsobiť softvér a nie naopak
- **nestálosť:** mení sa okolie a mení sa aj softvér (nie nahrádza novým), rozširujú sa požiadavky na úspešne používaný softvér, softvér prežíva hardvérové prostriedky
- **neviditeľnosť:** neexistuje akceptovateľný spôsob reprezentácie softvérového výrobku tak, aby sa pokryli všetky aspekty; dokonca nevieme ani určiť čo z príslušnej reprezentácie chýba

To, že vyššie uvedené podstatné problémy ohraničujú proces tvorby softvéru a nemožno ich úplne odstrániť neznamená, že ich nemôžeme zredukovať, t.j. zmenšiť ich dosah na kvalitu procesu tvorby softvéru a aj softvéru samotného. Napr. problémy spojené so zložitosťou môžeme zredukovať využitím princípov štruktúrovaného a modulárneho programovania, dekompozíciou problémov na podproblémy,...

Nie zákonité problémy – pravdepodobne možno vyriešiť a odstrániť

- **špecifikácia požiadaviek**
  - problémy s komunikáciou s používateľom
  - nejasná a neúplná formulácia požiadaviek spojená s neucelenou predstavou používateľa o výslednom softvérovom systéme
  - nejednoznačnosť spojená s častou špecifikáciou požiadaviek v prirodzenom jazyku
  - nestálosť požiadaviek
  - protirečivosť požiadaviek
  - prirodzená neúplnosť a nepresnosť pri špecifikácii veľkých softvérových systémov
  - nedostatok znalostí z analyzovanej oblasti a z toho vyplývajúce problémy s plánovaním softvérového projektu
  - ovplyvňovanie návrhu tvorbou špecifikácie a tým zanedbanie možných alternatívnych riešení
  - problémy s testovaním a verifikáciou špecifikácií
- **programátorská produktivita** (extrémne individuálne odchýlky, až 1:20)
- **slabá opakovateľnosť v tvorbe softvéru** (v procese tvorby softvéru je málo štandardizácie a väčšinou sa softvér tvorí vždy od začiatku, teda s každým programom sa vymýšľa už vymyslené; málo produktov sa zostavuje z už existujúcich súčiastok)
- **náchylnosť na softvéru chyby**
- **absencia "výroby" softvéru**
- **práca v tíme** (problémy s organizáciou práce v tíme pri veľkých softvérových projektoch)
  - komunikačné problémy, ktoré tvoria jeden z hlavných zdrojov programových chýb
  - problémy s plánovaním procesu tvorby softvéru
- **tvorba dokumentácie** (často sa porovnáva so samotným procesom tvorby programu)
  - enormné množstvo dokumentácie čo do kvantity aj rozmanitosti (napríklad vo veľkých vojenských softvérových projektoch bolo vytvorených 400 anglických slov na každý príkaz v programovacom jazyku Ada)
  - problémy s udržiavaním (modifikovaním) dokumentácie vzhľadom na meniacu sa programovú zložku softvéru
  - problémy s konzistentnosťou dokumentácie vzhľadom na aktuálny stav softvéru
  - problémy s úplnosťou dokumentácie

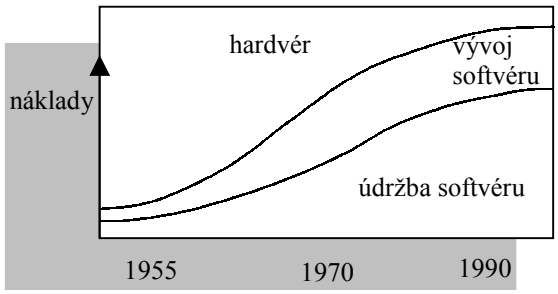
  Problémy s dokumentáciou sa najviac prejavujú pri prevádzke softvéru, keď pri údržbe softvéru sú nevyhnutné zmeny softvérového výrobku.
- **mnohé chyby, nedostatky sa spravidla objavujú až v prevádzke** (a nie v čase vývoja/výroby); ich odstraňovanie vedie k návratu k etapám vývoja softvéru
- **spôsob „starnutia softvéru"**
  - stála akumulácia prídavnej funkcionality v spojení s častými opravami chýb vedie k degradácii štruktúry a k zníženiu spoľahlivosti softvérových systémov s časom
  - softvér sa fyzicky neopotrebuje

Dôsledky

- nárast nákladov na vývoj a údržbu softvéru
- dodáva sa neskoro, nespoľahlivý a nie podľa požiadaviek (aj cenových a časových)

Problém mierky

- metódy použiteľné na riešenie malých problémov sa nedajú prispôsobiť na riešenie veľkých (zložitých) problémov



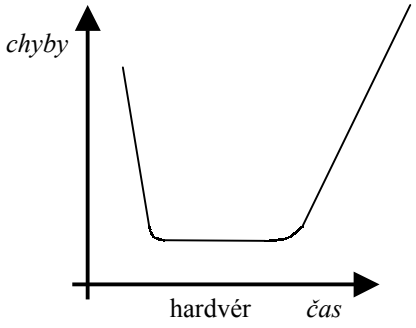
Najčastejšie príčiny zastavenia softvérových projektov

podľa analýzy viac ako 350 firiem a 8000 aplikácií (Pfleeger, 1998)

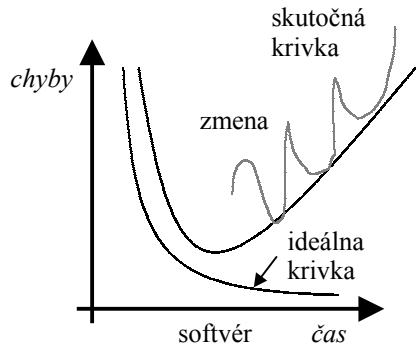
- neúplnosť alebo nejasnosť požiadaviek (13.1%)
- nedostatok záujmu a podpory zo strany používateľa (12.4%)
- nedostatok zdrojov, t.j. podhodnotený rozpočet a krátke termíny (10.6%)
- nerealistické očakávania (9.9%)
- nedostatočná podpora zo strany manažmentu dodávateľa alebo odberateľa (9.3%)
- zmena požiadaviek a špecifikácie (8.7%)
- nedostatočné plánovanie (8.1%)
- už nie je potreba vyvíjaného systému (7.5%).

Starnutie hardvéru a softvéru

Krivka počtu chýb v závislosti od času pre iné inžinierske produkty (napr. hardvér) ukazuje, že na začiatku je pomerne veľa chýb, tie sa odstránia a v určitom čase je počet chýb relatívne stabilný. Neskôr počet chýb znovu rastie, jednotlivé súčiastky sa kazia, systém sa opotrebuje.



Ideálnu krivku pre softvérové systémy znázorňuje spodný obrázok. Vzhľadom na zmeny v softvérovom systéme skutočná krivka má iný priebeh. Po každej zmene počet chýb vzrastie a skôr než sa ustáli zvyčajne dôjde k ďalšej zmene. Softvér sa neopotrebuje, ale časom sa zhoršuje.



Dôležité faktory úspechu softvérových projektov

- zainteresovanosť používateľov (18%)
- podpora manažmentu používateľa (16%)
- jasne definované požiadavky (15%)
- dobré plánovanie (11%)
- realistické očakávania (9%)
- správna dekompozícia úlohy (9%)
- kompetentnosť zúčastnených (8%)

História softvérového inžinierstva

Šesťdesiate roky:

- ťažkosti spojené s vývojom väčších programov;
- zavedenie pojmu "softvérové inžinierstvo" na konferenciách v rokoch 1968-1969 spolu s pojmom "softvérová kríza";
- hľadanie jednoduchého a účinného riešenia na existujúce ťažkosti vyústuje do štruktúrovaného programovania, ktorého začiatok sa zvykne stotožňovať s Dijkstrovým článkom o používaní príkazu GOTO.

Začiatok sedemdesiatych rokov:

- snahy o prekonanie softvérovej krízy vedú cez výskum „dobrých" programovacích praktík;
- zdôrazňovanie ľudských faktorov v programovaní;
- rozpoznanie výhod návrhu zhora nadol, postupného zjemňovania a modulárneho programovania;
- zavedenie nových jazykov (vrátane Pascalu) a nových techník programovania v tímoch (pravidlo vedúceho programátora);
- jednoduché techniky štruktúrovaného programovania sa nahrádzajú metodológiami: štruktúrovanou analýzou, návrhom;
- uvedomenie si životného cyklu tvorby softvéru;
- podpora manažmentu tvorby softvéru;
- snahy o zaistenie akosti spôsobujú vývoj procedúr zameraných na systematické testovanie, zavedenie pojmov formálnej správnosti programu.

Sedemdesiate roky:

- snahy o automatizáciu úlohy syntézy programov vedú k návrhu niekoľkých metód schopných automatickej syntézy programov (veľkosti programov nepresahujú niekoľko riadkov);
- využitie deduktívneho, induktívneho a iných transformačných prístupov pri realizácii formálnej transformácie špecifikácie do programu;
- používanie abstrakcie a modulárnej dekompozície ako návrhových techník;
- rozpracovanie pojmu abstraktných dátových typov, ktoré intenzívne pokračuje aj v 80-tych rokoch;
- ďalší rozvoj metodológií, pričom sa vyhraňujú dátovo a procesne orientované metódy;
- uvedomenie si významu a dôležitosti etáp špecifikácie a návrhu;
- prvé princípy zo začiatku 70-tych rokov sa začínajú široko používať v počítačovom priemysle.

Osemdesiate roky:

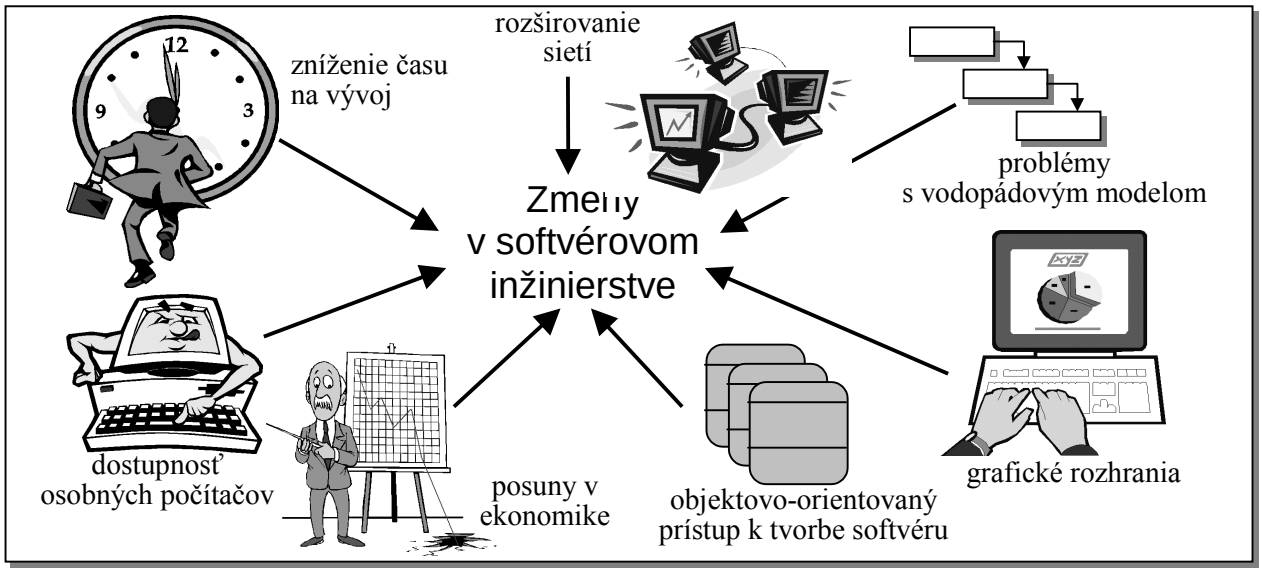
- rozšírenie používania programovacích prostredí;
- snaha o počítačovú podporu jednotlivých metód s čím súvisí vývoj CASE prostriedkov;
- vývoj nových programových paradigiem ako objektovo-orientované programovanie;
- ďalší vývoj funkcionálneho, logického ako aj imperatívneho programovania;
- vývoj formálnych metód špecifikácie a návrhu väčších programov;
- pokroky v oblasti paralelného programovania;
- zvýšenie pozornosti etape prevádzky a údržby softvéru, dôsledkom čoho je vývoj systémov na podporu údržby verzií softvérových objektov a riadenia konfigurácií softvérových systémov.

Deväťdesiate roky:

- rozšírenie prototypovania;
- vývoj softvéru na základe znovupoužiteľnosti (angl. reusability) a komponentov (najmä v súvislosti s objektovo-orientovaným prístupom k tvorbe softvéru);
- ďalší vývoj objektovo-orientovaného programovania, rozšírenie jazyka Java;
- pozornosť sa venuje objektovo-orientovanej špecifikácii a návrhu softvérových systémov, definujú a používajú sa schémy a vzory (napr. návrhové vzory)
- aplikácia techník znalostných systémov a umelej inteligencie do softvérového inžinierstva;
- sledovanie akosti softvérového procesu a softvéru použitím metrík;
- vývoj otvorených softvérových systémov, často s otvoreným zdrojovým textom programu (angl. open source software);
- snahy o efektívne využitie rozširujúcej sa celosvetovej siete vyúsťujú do nových metód, techník a prostriedkov spolupráce pomocou internetu a intranetu; veľká pozornosť sa sústreďuje na tvorbu distribuovaného softvéru a na metódy a techniky distribuovanej tvorby softvéru.

Prečo je vývoj softvéru v deväťdesiatych rokoch iný ako 10 alebo 20 rokov dozadu (Wasserman, 1996)

1. zníženie času na vývoj softvéru (komerčné výrobky)
2. posuny v ekonomike (zníženie ceny hardvéru a zvyšovanie cien vývoja softvéru a údržby)
3. dostupnosť osobných počítačov (posun v zodpovednosti za vývoj, tabuľkové procesory)
4. rozširovanie sietí (dostupnosť informácií, nové požiadavky na aplikácie)
5. dostupnosť a používanie objektovo-orientovaného prístupu (znovupoužitie, komponentový prístup k vývoju softvéru)
6. grafické rozhrania (okná, ikony, menu,…)
7. problémy s vodopádovým modelom vývoja softvéru – potreba paralelnej práce a teda iného modelu; prototypovanie.

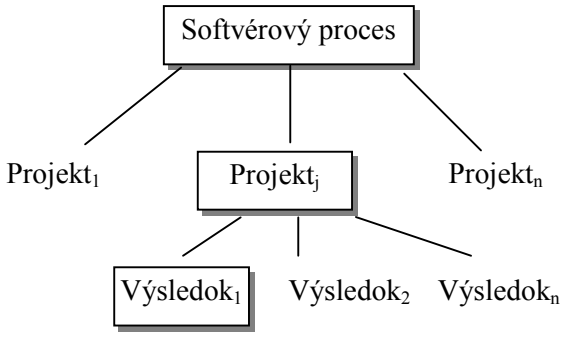


Softvérový proces a softvérový projekt

Softvérový proces – proces, zahŕňajúci technické aspekty a aspekty manažmentu vývoja softvéru; určuje abstraktnú množinu činností, ktoré sa majú vykonať pri vývoji softvérového výrobku z pôvodných požiadaviek používateľa.

Softvérový projekt – vykonanie týchto činností pre špecifické požiadavky používateľa, konkretizuje činnosti a poradie definované procesom (projektový plán). (Časovo ohraničené úsilie, ktoré sa vyvíja s cieľom vytvorenia jedinečného výsledku; množina činností, technických aj riadiacich, ktoré sa požadujú na zabezpečenie podmienok *projektovej dohody*.)

Výsledok – výstup vytvorený počas projektu



Softvérový projekt musí mať
- určený dátum začiatku a konca,
- dobre definované ciele a ohraničenia,
- stanovené zodpovednosti, rozpočet, rozvrh.

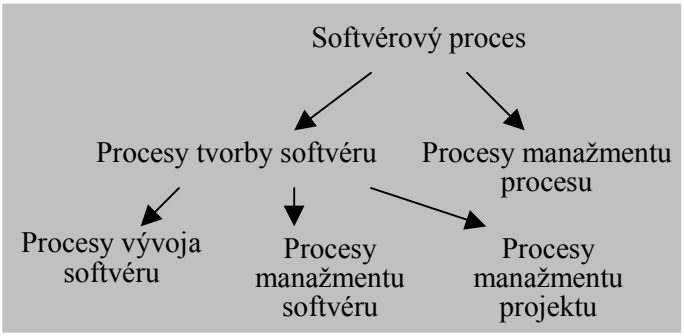
Softvérový projekt môže byť samostatný alebo môže byť časťou väčšieho projektu. V niektorých prípadoch softvérový projekt zahŕňa iba časť životného cyklu softvéru. Môže trvať veľa rokov a pozostávať z viacerých podprojektov, pričom každý z nich je dobre definovaný a samostatný.

Klasifikácia softvérových procesov

Procesy manažmentu procesu: zlepšovanie procesu = porozumenie existujúcim procesom a ich zmena tak, že sa zlepšia vlastnosti softvéru a/alebo sa redukujú náklady a čas na vývoj (akosť).

Proces vývoja softvéru: činnosti priamo spojené s vývojom softvéru – špecifikácia softvéru, realizácia softvéru, validácia softvéru a evolúcia softvéru.

Procesy manažmentu softvéru: (aj manažment softvérových konfigurácií) riadenie zmien softvérového systému, identifikácia jednotlivých verzií a konfigurácií počas jeho celého životného cyklu.

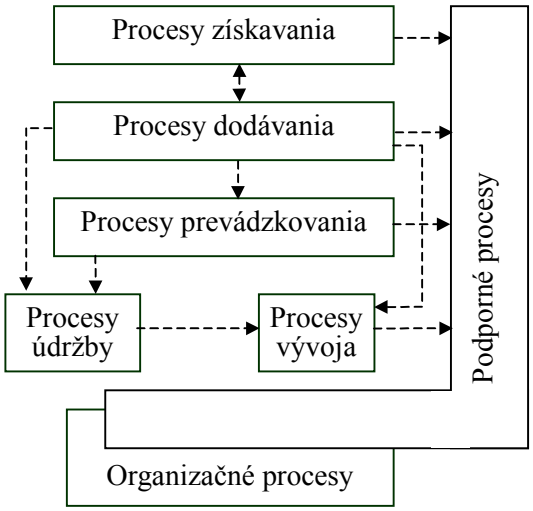


Procesy manažmentu projektu: použitie znalostí, zručností, prostriedkov a techník na projektové činnosti s cieľom dosiahnutia (alebo prekročenia) potrieb a očakávaní projektu.

ISO/IEC 12207, Štandard for Information Technology (Software life cycle processes, 1996)

Základné procesy:
- získavanie
- dodávanie
- vývoj
- prevádzkovanie
- údržba

Podporné procesy:
- dokumentovanie
- manažment softvérových konfigurácií
- zabezpečenie akosti
- verifikácia, validácia
- spoločné prehliadky
- audit
- riešenie požiadaviek a problémov



Organizačné procesy:
- manažment
- infraštruktúra
- zlepšovanie
- školenie

Charakteristiky softvérového procesu

Zrozumiteľnosť: do akej miery je proces definovaný explicitne a ako jasne je definovaný.

Viditeľnosť: sú výsledky procesu viditeľné zvonka?

Prijateľnosť: je definovaný proces prijateľný a použiteľný inžiniermi na tvorbu softvéru?

Spoľahlivosť: možno sa vyhnúť chybám (alebo ich dostatočne zavčasu zachytiť) tak, aby sa zabránilo chybám výsledku?

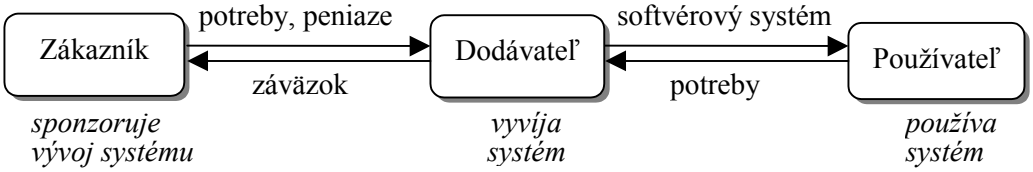
Robustnosť: môže proces pokračovať aj po výskyte neočakávaných problémov?

Udržovateľnosť: možno proces meniť, zlepšovať tak, aby odrážal zmenu požiadaviek?

Rýchlosť: ako rýchlo možno realizovať proces, ktorý z danej špecifikácie vytvorí výrobok (softvér)?

Podporovateľnosť: ako možno proces podporiť rôznymi softvérovými nástrojmi?

Účastníci tvorby softvéru



Používateľ /zákazník

☞ *Zákazník má vždy pravdu* ☞ *Zákazník platí (alebo neplatí).* ☞ *Komunikujte s používateľom!!!*

Rôzne skupiny:

- podľa činnosti (lokálny a globálny pohľad):
  - lokálny pohľad, zaujíma sa o funkčnú stránku, zaujíma ho fyzický pohľad na systém
  - nemusí mať lokálny pohľad, pozná výkonnú stránku, zaujíma sa o rozpočet, sprostredkovateľ medzi manažérmi
  - globálny pohľad, inicializácia projektu, nemá priame skúsenosti s výkonnou stránkou, strategické ciele
- podľa úrovne skúseností s výpočtovou technikou

Analytik

- vniknutie do problémovej oblasti, do ktorej spadá tvorba softvérového systému (napr. spracovanie informácii v organizácii používateľa)
- získať požiadavky od používateľa
- vyriešiť rôzne pohľady používateľov
- poradiť v otázkach čo je a nie je technicky možné
- overiť požiadavky
- zapísať požiadavky
- vyjednať súhlas s požiadavkami
- na základe požiadaviek používateľa sa podieľať na tvorbe špecifikácie softvéru a architektonického návrhu systému,
- definovať jednotlivé funkčné bloky a ich informačné prepojenia.

Analytik sa zapája do prác najmä v etapách analýzy a špecifikácie požiadaviek a architektonického. Úloha analytika je náročná, pretože musí riešiť problémy, na ktoré neexistuje jednoznačná a jediná správna odpoveď. Jeho prácu ovplyvňujú externé okolnosti.

Jadro prace analytika spočíva v intenzívnej komunikácii s používateľmi. Z tohoto dôvodu musí mať bohaté skúsenosti v komunikácii s ľuďmi. Rovnako sa vyžadujú znalosti s prace s podpornými prostriedkami, napr. CASE systémami.

Návrhár

- vytvoriť špecifikáciu softvéru
- návrh architektúry systému
- tvorivá práca, ktorá má významný dosah na budúce fungovanie softvéru pri údržbe

Úlohou návrhára je vytvorenie špecifikácie softvéru (počítačového riešenia problému). Návrhár je aktívny najmä v etape architektonického návrhu a podrobného návrhu. Návrhár musí vyberať z mnohých alternatív.

Návrhár musí mať aj skúsenosti z programovania, aby vedel zhodnotiť dôsledky výberu istého spôsobu riešenia na vlastnosti výsledného systému. Tiež sa často vyžadujú skúsenosti v práci s CASE systémami.

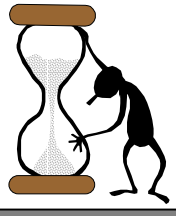
Programátor

Úlohou programátora je implementácia navrhnutého riešenia. Má znalosti najmä z oblasti programovacích jazykov, podporných prostriedkov a operačných systémov. Práca programátora je presná a kontrolovateľná: program alebo vyhovuje špecifikácii, alebo nie.

Často sa úlohy analytika, návrhára a programátora spoja. V tomto prípade analýzu, návrh a implementáciu vykonáva ta istá osoba. Takýto prístup nie je akceptovateľný pri väčších projektoch, pretože ohraničuje možné alternatívy skorým zohľadňovaním možných implementačných problémov.

Ako trávia čas programátori? (Jalote, 1997)

Písanie programov	13%
Čítanie programov a príručiek	6%
Komunikácia týkajúca sa práce	42%
Ostatné (vrátane osobných)	39%



Odborník na údržbu

- „údržbár“ (angl. maintenance programmer)
- náročná úloha, vykonáva činnosti analytika, návrhára aj programátora

Odborník na testovanie

Prvé testovanie pri vývoji programu (softvérovej súčiastky) vykonávajú samotní programátori. Konečné testovanie softvérovej súčiastky by však nemal vykonávať jeho tvorca a navyše, toto testovanie by sa malo vykonávať systematicky, aby sa testovaním pokryla čo najväčšia časť systému.

Testovanie je pomerne zložitá oblasť. Z tejto zložitosti však vyplýva, že testovaniu by sa mal venovať špeciálny odborník, nie programátor.

Podporný personál

- operátori
- technici

Manažment

Manažment riadi, zabezpečuje naplnenie požiadaviek, ktoré definuje vývojový tím podľa požiadaviek zákazníka a budúceho používateľa na to, aby úspešne vyvinul požadovaný softvér:

- čím vyššie postavený, tým menej vie o počítačovej technike (všeobecne)
- ciele manažmentu môžu byť v rozpore s cieľmi používateľa
- rôzne pohľady rôznych manažérov.

Auditor, skupina na zabezpečenie akosti

angl. software quality assurance (SQA) group

- väčšinou vstupujú do projektu neskôr
- často sa do tejto skupiny zahŕňajú aj odborníci na testovanie
- sústreďujú sa najmä na sledovanie a zabezpečenie akosti vytváraného výsledku, ale aj procesu
- zabezpečujú aj manažment softvérového systému (riadenie zmien, správu verzií a konfigurácií).