Časť I – Softvér a inžinierstvo

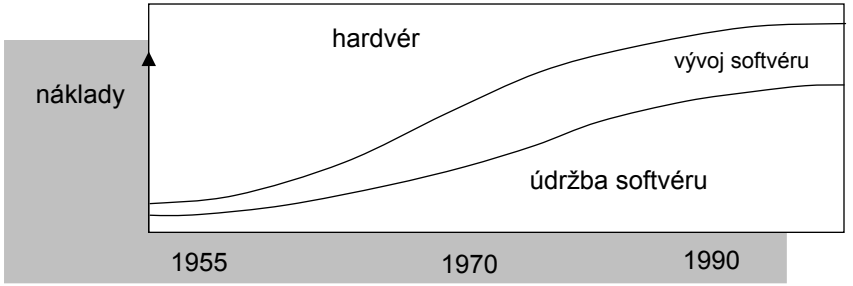
Čo to je softvérové inžinierstvo?

- ??
-
-
-

Softvérové inžinierstvo ≠ programovanie!!!!

Prečo je štúdium softvérového inžinierstva dôležité?

- ?
-
-



Zavedenie pojmu softvérové inžinierstvo

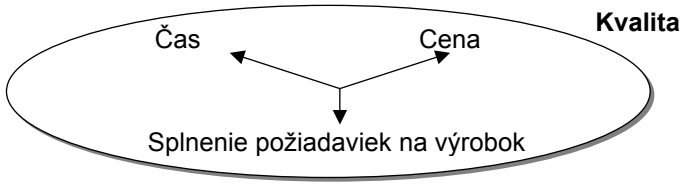
- zavedenie pojmu "softvérové inžinierstvo" na konferenciách v rokoch ???-??? spolu s pojmom "softvérová kríza",
- softvérové inžinierstvo má **?? rokov**

Poznámky:

Niekoľko dôležitých pojmov

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

- nie ako absolútna miera, ale ako stupeň splnenia požiadaviek, resp. potrieb.
- ak cieľom bude vyvinúť nespoľahlivý softvér, potom čím menej bude spoľahlivý, tým je kvalitnejší.



Kvalita je ...

- miera stupňa dokonalosti (Oxfordský slovník)
- splnenie požiadaviek (Crosby)
- vhodnosť k danému účelu (ISO 9001)
- schopnosť produktu alebo služby plniť dané potreby (BS 4778)

Metóda

- procedúra, postup ako dosiahnuť nejaký výsledok

Technika

- spôsob ako vykonať určité činnosti

Prostriedok

- nástroj, ktorý sa použije pri riešení problému

Metodológia

- súbor metód a techník predstavujúcich overený postup

Pozor na anglické pojmy!!!

História softvérového inžinierstva

60-te 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 vyúsťuje do **štruktúrovaného programovania**, ktorého začiatok sa zvykne stotožňovať s Dijkstrovým článkom o používaní príkazu GOTO.

70-te roky

Začiatok 70-tych 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);
- uvedomenie si **životného cyklu tvorby softvéru**;
- jednoduché techniky štruktúrovaného programovania sa nahrádzajú **metodológiami**: štruktúrovanou analýzou, návrhom;
- podpora manažmentu tvorby softvéru;
- snahy o **zabezpečenie kvality** spôsobujú vývoj procedúr zameraných na systematické testovanie, zavedenie pojmov formálnej správnosti programu.

Koniec 70-tych rokov

- 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.

80-te 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.

90-te 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 kvality/vyspelosti **softvérového procesu** a softvéru použitím **metrík**;
- vývoj **otvorených softvérových systémov**, často s otvoreným zdrojovým textom programu (angl. open source software);
- snahy o efektívne využitie **Internetu**; nové metódy, techniky a prostriedky spolupráce; pozornosť sa venuje tvorbe distribuovaného softvéru a metódam a technikám distribuovanej tvorby softvéru.

Začiatok 21. storočia

- ???????

Hľadajte o histórii informácie na webe!!!

Nové skutočnosti v tvorbe softvéru

1. zníženie času na vývoj softvéru (komerčné výrobky)
2. zvyšovanie zložitosti softvérových systémov
3. posuny v ekonomike (zníženie ceny hardvéru a zvyšovanie cien vývoja softvéru a údržby)
4. rozširovanie používania softvéru v najrôznejších oblastiach ľudskej činnosti
5. nové požiadavky na softvér (vnorené systémy, mobilné systémy, adaptivita, neviditeľné počítanie)
6. dostupnosť osobných počítačov (posun v zodpovednosti za vývoj)
7. rozširovanie sietí (dostupnosť informácií, distribuované aplikácie, distribuovaná tvorba softvéru)
8. dostupnosť a používanie objektovo-orientovaného prístupu (znovupoužitie, komponentový prístup k vývoju softvéru, multiagentové systémy)
9. grafické rozhrania (okná, ikony, menu,…)
10. problémy s vodopádovým modelom vývoja softvéru – potreba paralelnej práce a teda iného modelu; prototypovanie

Softvérový produkt

Členovia jednej kultúry vytvárajú veci pre členov inej kultúry.

Softvér vs. program

– softvérový systém
– softvérový produkt
– softvérový výrobok

– 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, …

Prečo vlastne vytvárame softvér?

– ??
– ?
– ?
– ?

Typy softvérových výrobkov

– **generické:** softvér sa predáva ľubovoľnému záujemcovi (off the shelf),
COTS – commercial-of-the-shelf,
MOTS – modified-of-the-shelf
– **zákaznícke (na objednávku):** softvér sa vytvára na základe požiadaviek pre konkrétneho zákazníka.

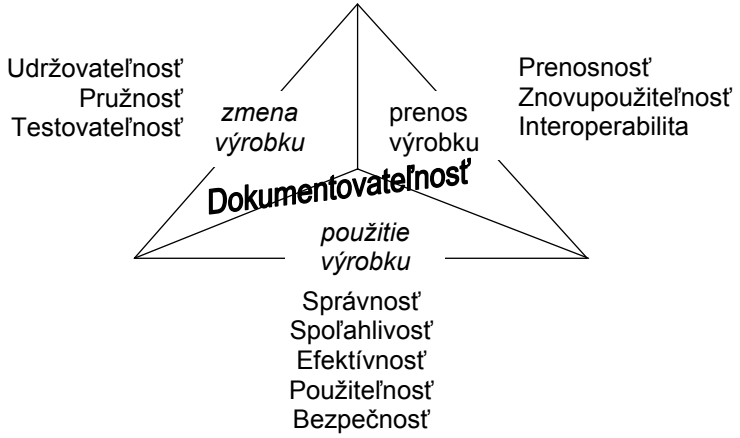
Typy softvérových aplikácií

– ???

Vlastnosti softvéru

- vyžadujú sa akonáhle sa softvér začne používať. Nejde o služby (funkcie), ktoré softvér zabezpečuje (vykonáva).

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.

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

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

Bezpečnosť: miera odolnosti voči neoprávneným zásahom do systému.

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.

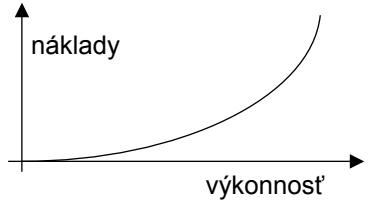
Pružnosť, modifikovateľnosť: úsilie, ktoré treba na modifikáciu výrobku v prevádzke (napr. zvýšenie jeho funkcionality).

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

Dokumentovateľnosť: miera, do akej sú všetky rozhodnutia počas vývoja zdokumentované a kontinuita dokumentácie počas všetkých etáp.

Vylučovanie sa niektorých vlastností navzájom

Vzťah medzi nákladmi na vývoj softvéru a výkonnosťou softvéru



Problémy s tvorbou softvéru

Podstatné, vnútorné problémy

- **zložitosť**
- **prispôsobivosť**
- **nestálosť**
- **neviditeľnosť** (*syndróm 90% hotovo*)

Nie zákonité problémy

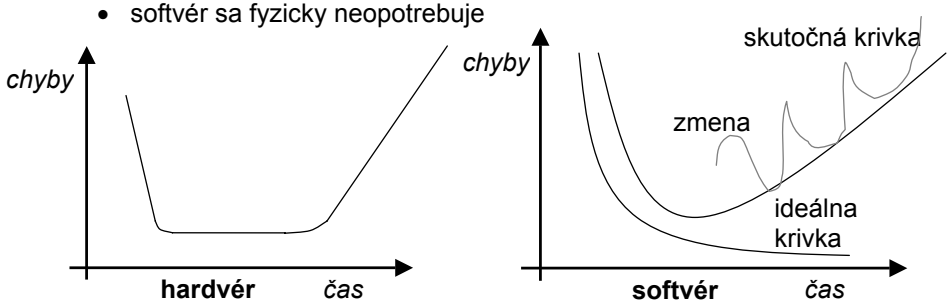
- špecifikácia požiadaviek
  - komunikácia s používateľom

- nejasná a neúplná formulácia požiadaviek, neucelená predstava používateľa o výslednom softvérovom systéme
- nejednoznačnosť spojená s častou špecifikáciou požiadaviek v prirodzenom jazyku
- nestálosť, protirečivosť požiadaviek
- prirodzená neúplnosť a nepresnosť pri špecifikácii veľkých softvérových systémov
- nedostatok znalostí z analyzovanej oblasti (problémy s plánovaním projektu)
- problémy s testovaním a verifikáciou špecifikácií

– **programátorská produktivita** (extrémne individuálne odchýlky, až 1:20)
– **slabá opakovateľnosť v tvorbe softvéru** (málo štandardizácie, väčšinou sa softvér tvorí vždy od začiatku; málo produktov sa zostavuje z už existujúcich súčiastok)
– náchylnosť softvéru na chyby
– absencia "výroby" softvéru
– **práca v tíme** (organizácia práce v tíme pri veľkých softvérových projektoch)
  - komunikačné problémy
  - plánovanie procesu tvorby softvéru
– **tvorba dokumentácie** (vs. samotný proces tvorby programu)
  - enormné množstvo dokumentácie čo do kvantity aj rozmanitosti (napr. vo veľkých vojenských softvérových projektoch bolo vytvorených 400 anglických slov na každý príkaz v programovacom jazyku Ada)
  - problémy s udržiavaním dokumentácie (meniaca sa programová zložka softvéru)
  - problémy s konzistentnosťou a úplnosťou dokumentácie

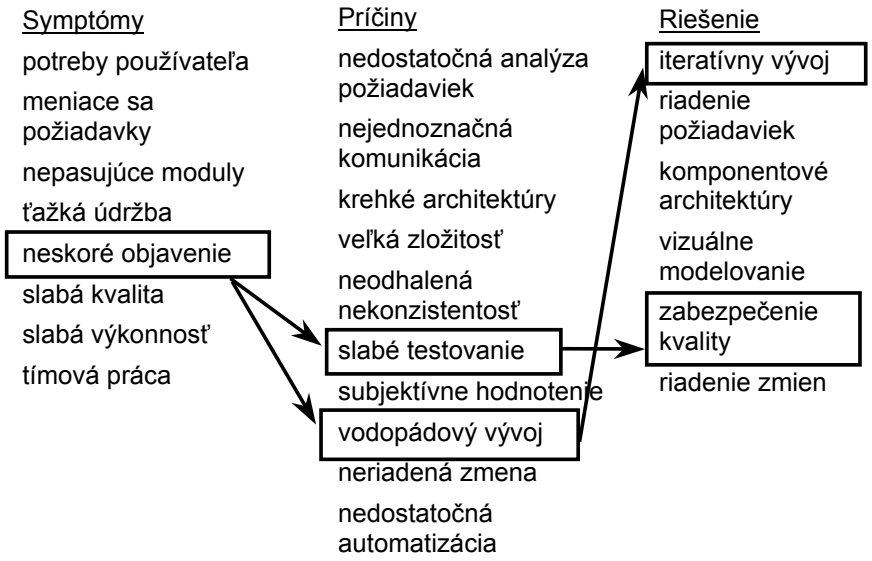
Problémy s dokumentáciou sa najviac prejavujú pri prevádzke softvéru, keď pri údržbe softvéru sú nevyhnutné zmeny softvérového výrobku.

– **mnohé chyby, nedostatky sa objavujú až v prevádzke** (návrat k predchádzajúcim etapám vývoja softvéru)
– **problém mierky**
– **spôsob „starnutia softvéru"**
  - stála akumulácia prídavnej funkcionality, časté opravy chýb → degradácia štruktúry a zníženie spoľahlivosti softvérových systémov s časom
  - softvér sa fyzicky neopotrebuje



Príčiny zastavenia softvérových projektov
Dôležité faktory úspechu softvérových projektov

Problémy s tvorbou softvéru – symptómy, príčiny, riešenia



Na okraj…

Ako vždy, motiváciou sú peniaze:

- Ako by mohli softvéroví inžinieri vyvinúť viac lepšieho a za menej peňazí?

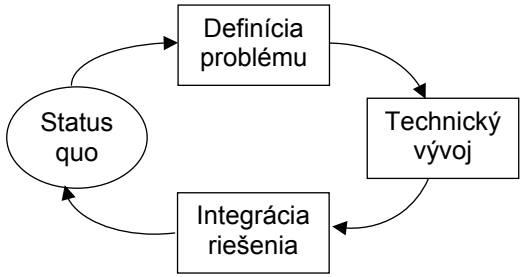
Vízia:

- Budeme schopní vyvíjať softvér spájaním hotových komponentov: rýchlo, spoľahlivo a lacno (product lines).

Proces vývoja softvéru

Vývoj softvéru – proces, v ktorom sa potreby používateľa transformujú na požiadavky na softvér, tieto sa transformujú na návrh, návrh sa implementuje v príslušnom programovacom jazyku, tento sa testuje a odovzdá používateľovi.

Je vývoj softvéru umenie alebo inžinierstvo?

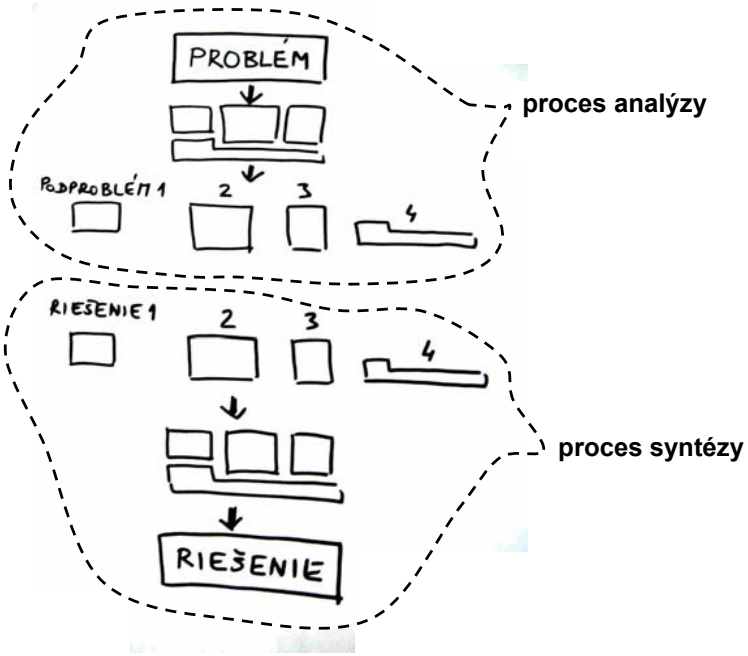


RIEŠENIE PROBLÉMU

Proces definuje

- kto robí
- čo,
- kedy a
- ako dosiahneme určitý cieľ.

Riešenie problémov



Životný cyklus softvéru

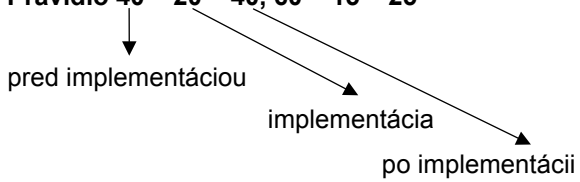
- rozdelenie (dekompozícia) zložitejšieho problému na jednoduchšie, ľahšie zvládnuteľné problémy
- rozdiely v existujúcich prístupoch

- podstatnou charakteristikou každého modelu životného cyklu je, že
  - **definuje jednotlivé etapy** a
  - pre každú z nich **činnosti**, ktoré sa majú vykonať,
  - rovnako ako vstupy a výstupy etapy

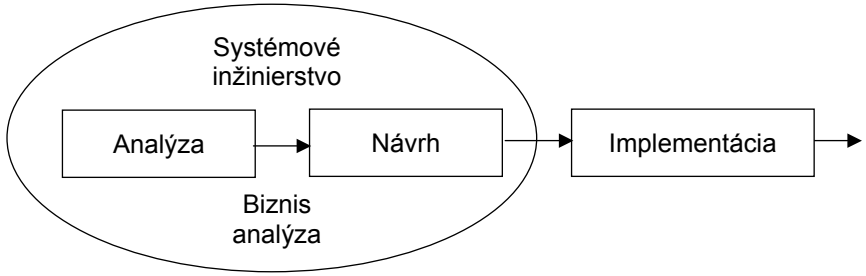
Činnosti priamo spojené s vývojom softvéru

- ??
-
-
-

Pravidlo 40 – 20 – 40, 60 – 15 – 25



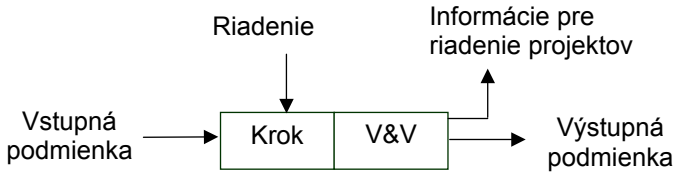
Systémové inžinierstvo a softvérové inžinierstvo



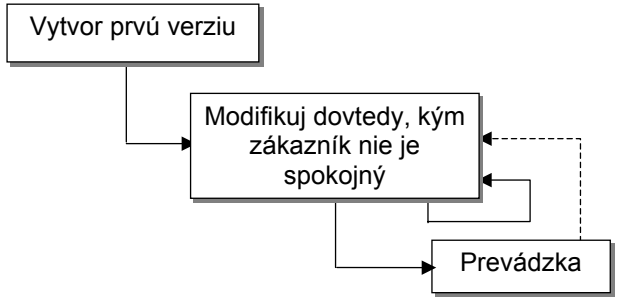
Model životného cyklu softvéru

- definuje jednotlivé činnosti (kroky), ktoré treba vykonať
- definuje časovú následnosť krokov
- **ne**definuje dĺžku trvania krokov a ich rozsah
- návrat k predchádzajúcim krokom
- odporúčania
- každá etapa dobre definovaná
- každá etapa vytvára „hmatateľné“ výstupy
- správnosť každej etapy možno vyhodnotiť

Znázornenie etáp

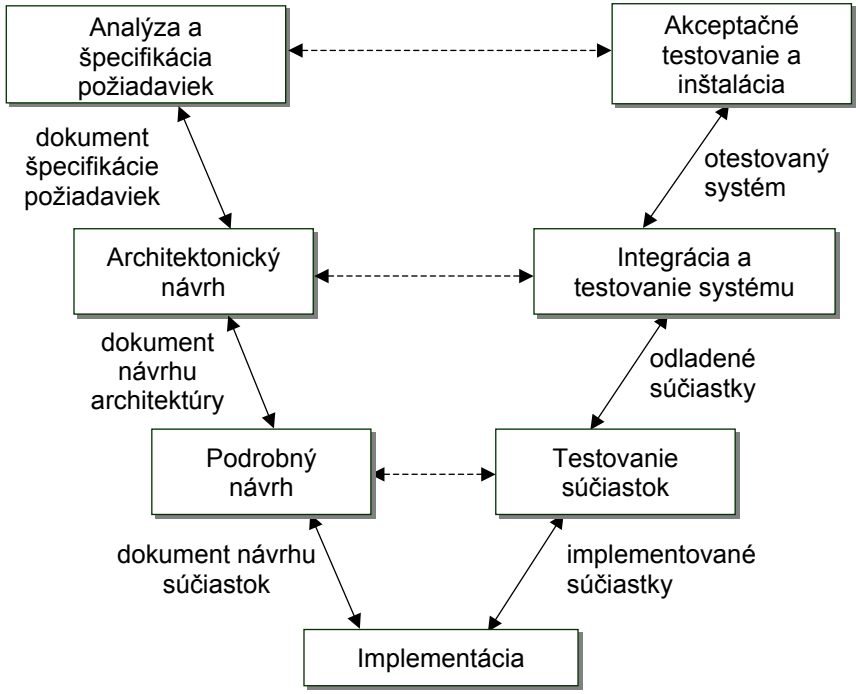


Model vytvor a oprav



V-model vývoja softvéru

- aspekt následnosti
- aspekt abstrakcie
- návratu k niektorej z predchádzajúcich etáp



Analýza a špecifikácia požiadaviek

- transformácia neformálnych požiadaviek používateľa do štruktúrovaného opisu týchto požiadaviek
- zdôraznenie **čo** chce používateľ a nie *ako* to možno zabezpečiť (realizovať)
- vykonanie štúdie vhodnosti
- získavanie, analýza, definovanie a špecifikácia požiadaviek
- plánovanie akceptačného testovania

Architektonický návrh

- ujasňuje sa celková koncepcia systému
- návrh dekompozície systému
- určenie vzťahov medzi časťami systému
- špecifikácia funkcionality a ohraničení pre každý podsystém
- plánovanie testovania systému
- plánovanie nasadzovania systému do prevádzky, dohoda o postupe nasadzovania podsystémov, dohoda o pláne zaškoľovania používateľov

Podrobný návrh

- podrobná špecifikácia softvérových súčiastok
- špecifikácia algoritmov realizujúcich požadované funkcie
- špecifikácia rozhraní pre jednotlivé súčiastky
- špecifikácia logickej a fyzickej štruktúry údajov, ktoré spracúva príslušná súčiastka
- špecifikácia spôsobu ošetrovania chybových a neočakávaných stavov
- plán prác pri implementácii súčiastky
- plán testovania súčiastky, návrh testovacích údajov
- špecifikácia potrebných ľudských zdrojov (odhad dĺžky trvania a nákladov)

Implementácia a testovanie súčiastok

- programová realizácia softvérových súčiastok
- vypracovanie dokumentácie k súčiastkam
- testovanie implementovaných súčiastok
- začiatok školení budúcich používateľov

Integrácia a testovanie systému

- spájanie súčiastok do podsystémov a systému
- testovanie podsystémov a celého systému
- integrácia podsystémov a systému
- testovanie podsystémov a systému (oprava nájdených chýb, časté návraty k etape implementácie)

Akceptačné testovanie a inštalácia

- testovanie systému používateľom
- preberacie konanie
- školenia použitia systému a nasadenie systému

Prevádzka a údržba

- zabezpečenie prevádzky softvéru
- riešenie problémov s používaním softvéru
- oprava, rozširovanie, prispôsobovanie softvéru podľa požiadaviek okolia

Známe modely životného cyklu softvéru

Vodopádový model

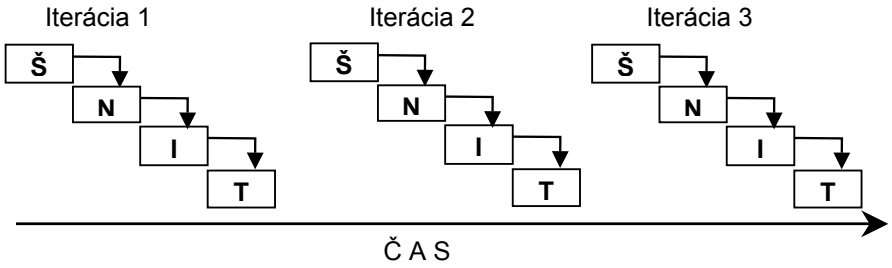
- nasledujúca etapa začne až po skončení predchádzajúcej

Inkrementálny (prírastkový) model

- systém sa vytvára a odovzdáva používateľovi po častiach (časti sa stanovujú na základe špecifikácie celého systému)

Iteratívny model

- systém sa vyvíja v iteráciách
- v každej iterácii sa vytvorí vykonateľný výsledok



Evolučný model

- požiadavky sa všetky nedefinujú na začiatku, systém sa vyvíja po častiach vo viacerých verziách

Špirálový model, WinWin model

- uvažuje aj aspekty manažmentu
- definuje množinu aktivít, ktoré smerujú k „výhre používateľa“ získaním produktu, ktorý zodpovedá jeho požiadavkám a „výhre vývojára“ vytvorením výsledku s reálnym rozpočtom a rozvrhom

Komponentový model

- využitie znovupoužiteľných súčiastok

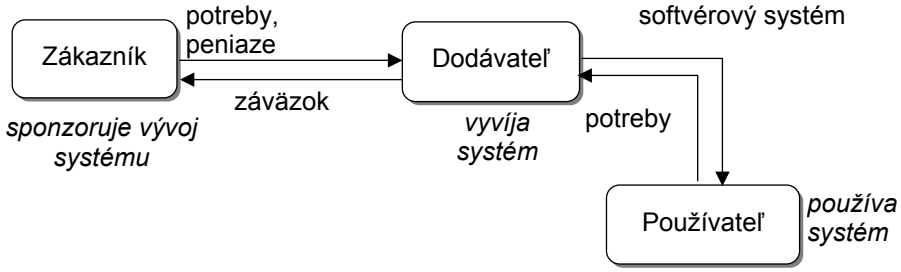
Formálna transformácia

- vyžaduje formálnu (matematickú) špecifikáciu systému

Agilné metódy tvorby softvéru

- eXtreme Programming, Feature Driven Development, Lean Programming, Crystal

Špecialisti v životnom cykle softvéru



Rôzne skupiny zákazníkov

- 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

Úlohy v softvérovom tíme

- ???
- ?

Ako trávia čas programátori?

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

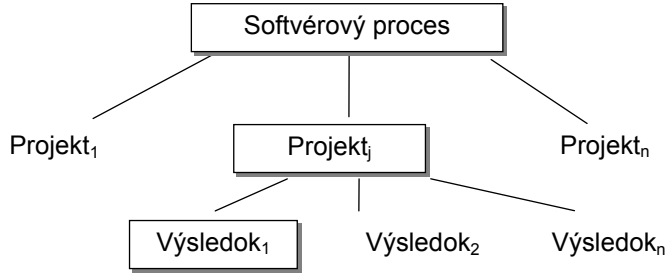
Softvérové procesy

Softvérový proces

- určuje abstraktnú množinu činností, ktoré sa majú vykonať pri vývoji softvérového výrobku z pôvodných požiadaviek používateľa.

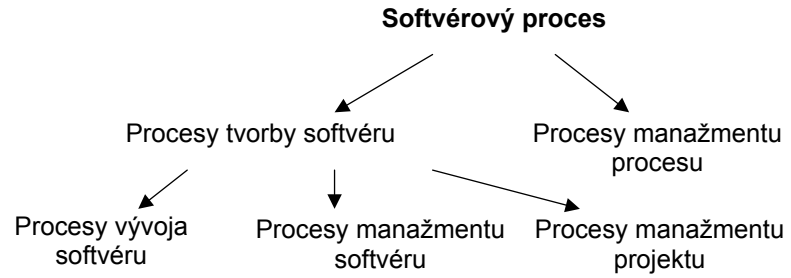
Softvérový projekt

- 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.*



Vyspelosť softvérového procesu – CMM

Klasifikácia softvérových procesov



Proces vývoja softvéru

- činnosti priamo spojené s vývojom softvéru – špecifikácia softvéru, realizácia softvéru, validácia softvéru a evolúcia softvéru.

Procesy manažmentu softvéru
(označuje sa 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.

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 (kvalita).

Etický kódex softvérového inžiniera

Výňatok z etického kódexu), plné znenie nájdete na http://computer.org/tab/SWECP9912.htm

Princíp 1: Verejnosť

Softvéroví inžinieri by mali konať v zhode so všeobecnými verejnými záujmami. Obzvlášť by mali:

- Prijať plnú zodpovednosť za vlastnú prácu.
- Schváliť softvérový produkt iba v prípade, ak sú odôvodnene presvedčení o tom, že produkt je bezpečný, spĺňa špecifikačné požiadavky, neznižuje kvalitu života, neohrozuje súkromie a životné prostredie a úspešne vyhovel príslušným testom.
- Prezentovať vhodnej osobe alebo autorite každé existujúce alebo potenciálne ohrozenie používateľa, verejnosti alebo životného prostredia, o ktorom sú odôvodnene presvedčení, že je spôsobené alebo spojené so softvérom alebo príslušnými dokumentmi.
- Byť povzbudení dobrovoľne poskytovať svoje profesionálne znalosti na dobré ciele a prispievať k zvyšovaniu úrovne znalostí verejnosti v oblasti softvérového inžinierstva.
- ....

Princíp 2: Zákazník a zamestnávateľ

Softvéroví inžinieri by mali jednať v najlepších záujmoch svojho zákazníka a zamestnávateľa, v zhode s verejným záujmom. Obzvlášť by mali:

- Vedome nepoužívať softvér, ktorý je získaný alebo prechovávaný nelegálne alebo neeticky.
- Identifikovať, dokumentovať, zhromažďovať údaje a oznámiť ich zákazníkom alebo zamestnávateľom okamžite, keď by podľa ich názoru

mohol byť projekt neúspešný, preukázateľne drahý, mohol by porušovať zákon duševného vlastníctva alebo byť ináč problematický.

- Neakceptovať žiadnu vedľajšiu prácu, ktorá by mala škodlivý vplyv na prácu, ktorú uskutočňujú pre svojho hlavného zamestnávateľa.
- Nepodporovať žiadne záujmy zamerané proti svojmu zamestnávateľovi alebo zákazníkovi, iba ak je to v zhode s vyšším etickým záujmom. V takom prípade by o ňom mali informovať zamestnávateľa alebo inú vhodnú autoritu.
- ....

Princíp 3: Výrobok

Softvéroví inžinieri by mali zaistiť, aby ich produkty a súvisiace modifikácie spĺňali najvyšší možný profesionálny štandard. Obzvlášť by mali:

- Usilovať o vysokú kvalitu, akceptovateľné náklady a zmysluplný plán, zabezpečiac, aby významné kompromisy boli jasné a akceptované zamestnávateľom aj klientom a prístupné stanovisku používateľa a verejnosti.
- Zabezpečiť svoju kvalifikáciu na každom projekte, na ktorom pracujú alebo sa spolupodieľajú, primeranou kombináciou vzdelania, školení a skúseností.
- Dodržiavať pri aktuálnej práci primerané profesionálne štandardy vždy, keď je to možné, odchyľujúc sa od nich iba v eticky a technicky oprávnených prípadoch.
- Zabezpečiť adekvátne testovanie a kontrolu softvéru a prislúchajúcich dokumentov, na ktorých pracujú.
- Zabezpečiť adekvátnu dokumentáciu, obsahujúcu objavené dôležité problémy a použité riešenia pre každý projekt, na ktorom pracujú.
- Prácou vytvárať softvér a prislúchajúcu dokumentáciu, ktorá rešpektuje súkromie tých, ktorí budú ovplyvnení daným softvérom.

- Pristupovať ku všetkým formám softvérovej údržby s rovnakou profesionalitou ako k novému vývoju.
- ...

Princíp 4: Posudzovanie

Softvéroví inžinieri by mali zachovávať nezávislosť vo svojom profesionálnom posudzovaní. Obzvlášť by mali:

- Dodržovať profesionálnu objektivitu so zreteľom na akékoľvek softvérové alebo súvisiace dokumenty pri požiadavke o ich zhodnotenie.
- Informovať všetky zúčastnené strany, ktorých konfliktom záujmov nemožno primerane zabrániť alebo sa im vyhnúť.
- ...

Princíp 5: Manažment

Manažéri a lídri softvérového inžinierstva by mali podporovať a propagovať etický prístup k manažmentu softvérového vývoja a údržby. Manažéri a vedúci softvéroví inžinieri by obzvlášť mali:

- Zabezpečiť dobrý manažment pre všetky projekty, na ktorých pracujú, vrátane efektívnych procedúr na podporu kvality a znižovania rizika.
- Pritiahnuť potenciálnych softvérových inžinierov iba úplným a presným opisom podmienok zamestnania.
- Umožniť riadny proces pri konaní o porušení koncepcie zamestnávateľa alebo tohto Kódexu.
- Nikoho netrestať za vyjadrenie etických pripomienok k určitému projektu.
- ...

Princíp 6: Profesia

Softvéroví inžinieri by mali zlepšovať bezúhonnosť a povesť profesie v súlade s verejným záujmom. Obzvlášť by mali:

- Zvyšovať povedomie verejnosti o profesii softvérového inžiniera.

- Rozširovať znalosť softvérového inžinierstva vhodnou účasťou v profesijných organizáciách, stretnutiach a publikáciách.
- Preberať zodpovednosť za hľadanie a opravu chýb, za vytváranie správ o chybách v softvéri a v príslušných dokumentoch, na ktorých pracujú.
- Oznámiť podstatné porušenia tohoto kódexu príslušným autoritám, keď je jasné, že konzultácie s dotknutými osobami sú nemožné, kontraproduktívne alebo nebezpečné.
- ...

Princíp 7: Kolegovia

Softvéroví inžinieri by mali byť spravodliví ku svojím kolegom a mali by ich podporovať. Obzvlášť by mali:

- Napomáhať kolegom v profesionálnom raste.
- Plne dôverovať práci ostatných a zdržať sa pripisovania nadmerných zásluh.
- Posudzovať prácu iných objektívne, poctivo a náležite zdokumentovaným spôsobom.
- Napomáhať kolegom k uvedomeniu si súčasných štandardných praktík vrátane bezpečnostnej politiky a procedúr na ochranu hesiel, súborov a ostatných citlivých informácií a bezpečnostné opatrenia všeobecne.
- V situáciách mimo svojho poľa pôsobenia požiadať o pomoc odborníka, ktorý má kompetencie v týchto oblastiach.
- …

Princíp 8: Osobnosť

Softvéroví inžinieri by sa mali zúčastňovať celoživotného vzdelávania týkajúceho sa ich profesnej praxe a mali by presadzovať etický prístup v profesnej praxi. Obzvlášť by sa mali softvéroví inžinieri usilovať:

- Zlepšovať svoju schopnosť vytvárať bezpečný, spoľahlivý a prospešný vysoko kvalitný softvér za rozumnú cenu a v rozumnom čase.

- Rozširovať svoje znalosti analýzy, špecifikácie, návrhu, vývoja, údržby a testovania softvéru a súvisiacich dokumentov spolu s manažmentom procesu vývoja softvéru.
- Zlepšovať svoju schopnosť vytvoriť presnú, informatívnu a dobre napísanú dokumentáciu.
- Zlepšovať svoje porozumenie softvéru a príslušnej dokumentácie, na ktorej pracujú a prostredia, v ktorom sa použijú.
- Zlepšovať svoju znalosť tohto Kódexu, jeho interpretácie a jeho aplikácie v praxi.
- Nezaobchádzať nespravodlivo s nikým z dôvodu akýchkoľvek irelevantných predsudkov.
- Nenabádať iných na vykonanie činov zahŕňajúcich porušenie tohto Kódexu.
- Poznať, že osobné porušenie tohto Kódexu je nezlúčiteľné s postavením softvérového inžiniera.

Na záver...

Fred Brooks: „How does a project really get into trouble?“
„One slip at a time.“

- nestačí zamestnať najlepších programátorov
- ani použiť najnovšie metódy, či technológie
- ani zapojiť používateľa v max. miere
- ani najať najlepšieho manažéra...

Treba urobiť toto všetko. A dokonca viac...

Softvérové inžinierstvo a ostatné inžinierske disciplíny

- „nehmotná“ povaha softvéru
- nemožnosť uvažovania všetkých podmienok, alternatív
- cieľom je minimalizácia škody pri neočakávaných podmienkach

Aspekty tvorby softvéru

- technický aspekt
- psychologický aspekt
- aspekt riadenia (manažmentu) projektu

Čo je pravda?

1. Na všetky činnosti pri tvorbe softvéru dnes existujú štandardy.
2. Najdôležitejšie je mať najnovšie CASE prostriedky a iné nástroje spolu s najnovšou výpočtovou technikou.
3. Keď sa oneskorujeme v softvérovom projekte v pláne, treba pridať viac programátorov do projektu, aby sme stratu vyrovnali.
4. Požiadavky na softvér sa neustále menia, ale keďže softvér je pružný (a nehmotný) možno ich jednoducho zapracovať.

5. Keď vytvoríme program, ktorý funguje – práca softvérového inžiniera skončila.
6. Pokiaľ nemám vykonateľný program, nemôžem o jeho kvalite nič povedať.
7. Výstup úspešného softvérového projektu je program.