Manažment softvérových systémov

- Činnosti pri manažmente softvérových systémov
- Verzie
  - *Typy verzií*
  - *Identifikácia verzií*
  - *Výber verzií*
  - *Uchovávanie verzií*
- Tvorba konfigurácie
- Modely spolupráce
  - *Optimistický model*
  - *Pesimistický model*
- Distribuovaná tvorba softvéru

Manažment softvérových systémov

Rozpor medzi

- snahou o zmrazenie stavu čiastkových výsledkov ako aj celého výrobku (pretože povaha ľudskej činnosti je diskrétna) a
- neustálym vývojom okolitého prostredia a z toho vyplývajúcej nutnosti zmien.

V literatúre sa manažment softvérových systémov označuje aj ako manažment softvérových konfigurácií.

Konfigurácia softvérového systému je množina softvérových komponentov spolu s ich definovanými vlastnosťami a vzťahmi slúžiaca na špecifický cieľ.

Najčastejšie používané vlastnosti komponentov

- systémové: *dátum, čas* (vytvorenia, modifikácie), *autor*, atď.
- súvisiace s vývojovým prostredím: *operačný systém*, *procesor*, *programovací jazyk*
- súvisiace so samotným procesom vývoja: *stav*, *opis zmeny*
- súvisiace s výkonnosťou: *rýchlosť*, *pamäť*
- súvisiace s riešeným problémom: *typ daňovej sústavy*, *použitý algoritmus*.

Činnosti pri manažmente softvérových systémov

Základné činnosti manažmentu konfigurácií sú (IEEE, 1990; ISO, 1995):

- *identifikácia* a dokumentácia charakteristík komponentov a verzií (konfigurácií) výrobku,
- *riadenie* zmien týchto charakteristík,
- *evidencia* a spracovanie stavu komponentov a celého výrobku,
- *overenie* úplnosti a konzistentnosti výrobku (konfigurácie).

Pokiaľ je predmetom manažmentu konfigurácií softvérový systém, hovoríme o manažmente softvérových systémov.

Špecifické vlastnosti softvérových systémov v porovnaní s inými inžinierskymi výrobkami vyžadujú modifikáciu (rozšírenie) ponímania manažmentu konfigurácií:

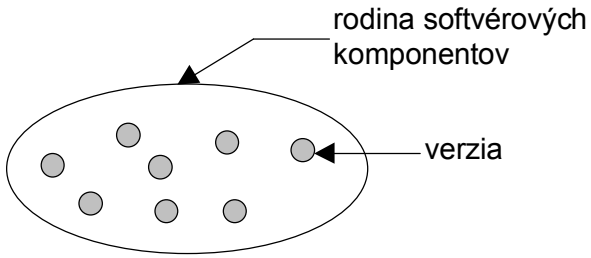
1. *Náchylnosť na chyby*
   Spôsobuje to najmä abstraktná povaha softvérových systémov (nejde o výrobky, ktoré možno „chytiť do ruky“), spôsob ich tvorby – nedostatočná znovupoužiteľnosť (iba ťažko zatiaľ možno hovoriť o výrobe softvérových systémov). Dôsledkom je zvýšenie možnosti zmien a tým aj zložitosti softvérového systému.
2. *Nerozlíšiteľnosť medzi originálom a duplikátom*
   Vzhľadom na počítačovú reprezentáciu a kvalitu dnešných kopírovacích strojov môžu existovať rôzne kópie softvérového systému (alebo jeho častí), napr. kópie patriace jednotlivým členom vývojového tímu, verejné kópie, kópie pre jednotlivých zákazníkov, atď. Tieto sa všetky môžu vyvíjať a teda meniť, čím môže nastať problém identifikácie správnej verzie.
3. *Rôznorodosť*
   Ide o rôznorodosť nielen problémových oblastí, pre ktoré sa vytvárajú softvérové systémy, ale aj samotných prostredí, vývojárskych tímov a podmienok jednotlivých organizácií. Dôsledkom je potreba častej zmeny procesu tvorby softvérového systému, najmä jeho štruktúry. Síce existujú viaceré normy súvisiace s tvorbou softvéru, väčšinou ich treba výrazne rozpracovať na aktuálne podmienky softvérového projektu.
4. *Počítačová reprezentácia*
   Všetky komponenty softvérového systému sú reprezentované v počítači. To ďalej umožňuje integráciu prostriedkov podporujúcich manažment softvérových systémov do počítačom podporených vývojových prostredí (keďže sa softvérové systémy tiež vyvíjajú na počítači).
5. *Distribuovaná tvorba*

Rozšírenie činností pre manažment softvérových systémov:

- identifikácia a dokumentácia charakteristík softvérových komponentov a konfigurácií softvérového systému;
- riadenie zmien týchto charakteristík;
- evidencia a spracovanie stavu komponentov a celého softvérového systému (konfigurácie);
- overenie úplnosti a konzistentnosti konfigurácie softvérového systému;
- *riadenie práce v tíme z pohľadu vývoja softvérového systému;*
- *riadenie výroby konfigurácie softvérového systému.*

Verzie

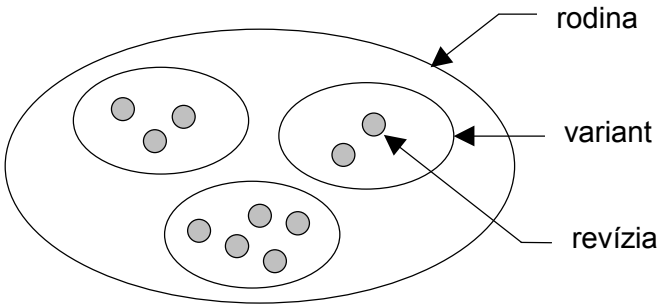
Vznikajú ako dôsledok rozporu medzi snahou o zmrazenie stavu a potrebou zmien.



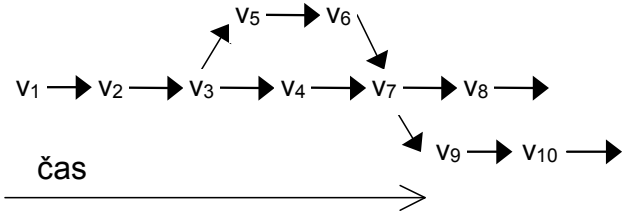
Typy verzií

Na základe príčiny vytvorenia verzie rozlišujeme medzi viacerými typmi verzií:

1. **Varianty:**
   - reprezentácia alternatívnych riešení alebo transformácií
   - navrhujú sa tak, aby existovali súčasne; žiadna z nich nie je hlavná.
2. **Revízie:**
   - oprava chýb
   - rozširovanie funkčnosti systému
   - prispôsobovanie zmenám prostredia
   - dôležitá je časová následnosť a má zmysel hovoriť o predchodcoch a/alebo nasledovníkoch verzie (nielen z časového hľadiska).
3. **Dočasné varianty:**
   - vznikajú na základe spolupráce (keď viacerí členovia pracovného tímu súčasne pracujú na vývoji jednotlivých komponentov)
   - neskôr dôjde k spojeniu takto vytvorených verzií
   - môžu vznikať aj napr. pri experimentálnom vývoji alebo pri požiadavkách na rozšírenie, vylepšenie, zavedenie nových vlastností systému, pričom vývoj pokračuje paralelne aj na pôvodnej verzii.



Často sa rozhoduje o type verzie (variant, revízia, dočasný variant) na základe histórie vývoja, ktorá sa reprezentuje *grafom histórie verzií.*



Identifikácia verzií

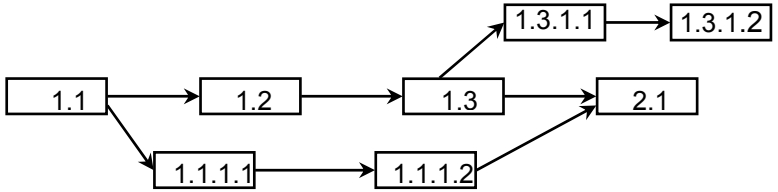
Verzie musia byť jednoznačne identifikovateľné v rámci rodiny softvérových komponentov. Treba pri tom rozlišovať, či ide o identifikáciu

- z pohľadu podporného prostriedku alebo
- z pohľadu používateľa.

Spôsob identifikácie významne ovplyvňuje model verzií.

Explicitné verziovanie (rodina verzií sa identifikuje explicitne vymenovaním verzií):

- vychádza sa z grafu histórie verzií a pojmy variant, revízia a dočasný variant sa odvodzujú od umiestnenia verzie v grafe
- revízie sa identifikujú jednoduchým číslom (ClearCase, Visual SourceSafe, DSEE) alebo dvojicou čísel – číslo zverejnenej verzie a číslo revízie (SCCS, RCS)



- varianty sa identifikujú napr. symbolickým menom (DSEE), hodnotami špecifických vlastností (Adele, Shape), cestou v grafe histórie verzií (ClearCase, Gypsy).

Implicitné verziovanie (verzie definujú predikátom a vytvárajú sa na požiadanie; takýto model sa často kombinuje s modelom založeným na zmene - verzia sa určí nepriamo logickým výrazom, pomocou ktorého sa identifikujú zmeny; aplikáciou týchto zmien sa vytvorí verzia)

- na identifikáciu zmeny sa používajú podobné techniky ako na identifikáciu explicitnej verzie.

Výber verzií

- použitie podmienok na vlastnosti verzií (často sa reprezentujú logickým výrazom, spolu s rôznymi rozšíreniami ako napr. umožnenie štandardného výberu, podmienkového výberu, zavedenie trojhodnotovej logiky)
- výber založený na heuristikách

Výber ako aplikácia postupnosti filtrov:

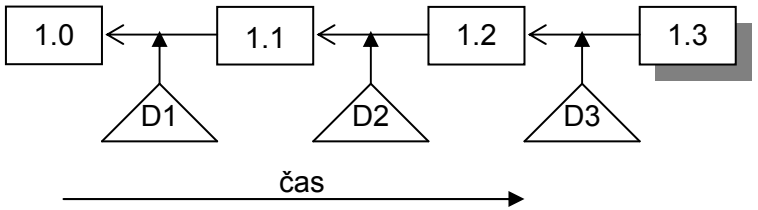
- výber sa realizuje ako postupná aplikácia jednotlivých podmienok (heuristík, filtrov) na množinu možných verzií
- ak po aplikácii niektorej z podmienok vznikne prázdna množina, treba sa vrátiť k predchádzajúcej množine možných verzií a aplikovať nasledujúcu, t.j. heuristika vedúca k prázdnej množine sa neuvažuje.
- podmienky výberu fungujú ako filtre a teda verzia sa vyberie aj v prípade, ak sa niektorá z nich nesplní.

Uchovávanie verzií

Rozdiel medzi dvoma verziami býva väčšinou „malý“ (menší ako ľubovoľná z oboch verzií). Toto sa používa pri uchovávaní verzií, keď sa namiesto celej verzie uchová iba rozdiel. Takýto spôsob uchovávania sa nazýva *delta technika*.

Pri použití delta techniky sa treba zaoberať tým

- ako sa vytvorí delta medzi dvoma verziami (a tiež ako sa znovu vytvorí verzia z delty a verzie uchovanej ako celok),
- ako aplikovať deltu na uchovanie komponentov v rámci rodiny softvérových komponentov, t.j. ktorú verziu uchovať v plnom tvare a vzhľadom ku ktorej verzii treba vypočítať deltu.



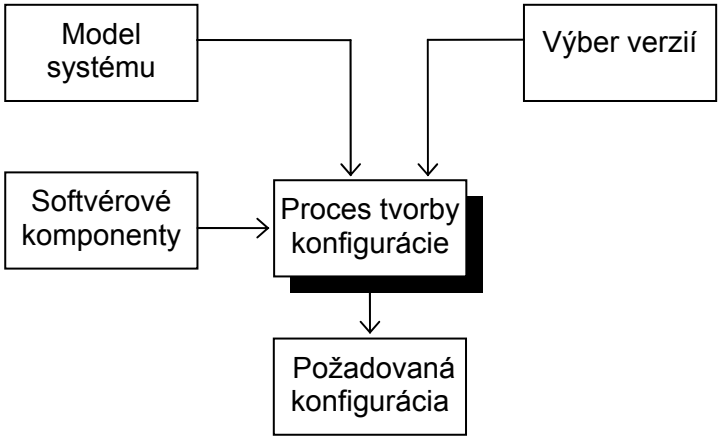
Pri vytváraní delty má veľký význam zvolená granularita rozdielov:

- riadok
- bit
- položka v databáze
- lexikálna jednotka

Použitie delta techniky je klasickým kompromisom medzi nárokmi na priestor a čas (vytvorenia a sprístupnenia verzie).

Delta techniku možno použiť nielen na riešenie (úsporného) uchovávania softvérových komponentov. Ďalšie možnosti vznikajú napr. rozširovaním služieb Internetu. V súvislosti s čoraz vyššími požiadavkami na dobu odozvy sa použitím delt výrazne znižujú nároky na objem prenášaných údajov.

Tvorba konfigurácie



Proces tvorby konfigurácie softvérového systému:

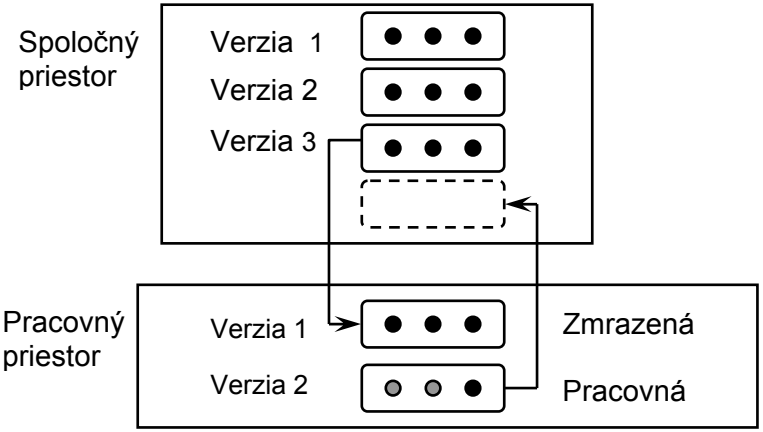
1. *Určenie množiny komponentov*, ktoré patria do konfigurácie
2. *Získanie komponentov konfigurácie* na základe množiny vytvorenej v prvom kroku. Tento krok zahŕňa aj vytvorenie odvodených komponentov, ak nie sú k dispozícii alebo nie sú aktuálne.

Pri riešení problému tvorby konfigurácie možno identifikovať niekoľko miest, v ktorých rozhodnutie o spôsobe riešenia ovplyvní proces tvorby konfigurácie:

- určenie komponentov, ktoré budú patriť do konfigurácie; možno ich vymenovať explicitne, alebo určiť implicitne pomocou ich vlastností
- existencia verzií; jednotlivé komponenty (rodiny) majú viaceré inštancie (verzie) s rôznymi charakteristikami
- požadovaný stupeň uspokojenia požiadaviek môže tiež spôsobiť rôzne variácie vzniknutej konfigurácie.

Modely spolupráce

- uplatňujú sa pri paralelnej práci v tíme
- používa sa pracovný priestor (angl. workspace), ktorý umožňuje izoláciu od zmien ostatných členov tímu



- súborový systém (komponenty sa vkladajú a vyberajú z podporného systému)
- databáza
- „virtuálny súborový systém“

Optimistický model

- predpokladá sa, že paralelná práce nevedie zvyčajne ku konfliktom
- štandardne umožňujú súčasnú prácu na jednotlivých komponentoch (verziách); zmeny sa integrujú neskôr

Pesimistický model

- uvažuje o možnosti konfliktov, poskytuje prostriedky na zaistenie, aby jednotlivé komponenty v danom okamihu mohol modifikovať iba jeden softvérový inžinier
- používa sa uzamykanie verzií komponentov (RCS): operácie check-in a check-out

Distribuovaná tvorba softvéru

Centrálny server

- klient/server architektúra; na serveri sú uložené všetky komponenty aj ich verzie;
- všetky operácie závisia od dostupnosti servera

Šírenie zmien v jednotlivých distribuovaných pracoviskách

- jednotlivé miesta zdieľajú spoločnú základnú verziu (baseline) a posielajú zmeny (namiesto verzií)

Individuálny pracovný priestor pre každé pracovisko

- každé pracovisko má v pracovnom priestore vetvy, ktoré reprezentujú ostatné pracoviská
- vetvy sa pravidelne aktualizujú (spájanie verzií)

Použitie distribuovanej databázy

- umožňuje transparentný prístup k verziám z ľubovoľného pracoviska