Techniky implementácie („Implementation Guidelines“)

Existujú externé a interné faktory kvality.
Externé: správnosť, robustnosť, flexibilnosť, znovupoužiteľnosť, kompatibilnosť, efektívnosť, portabilnosť, jednoduchosť používania
Interné: rôzne, napr. modulárnosť a zrozumiteľnosť.
Zaujímavé sú v konečnom dôsledku externé faktory, ale cesta k nim vedie cez interné faktory kvality.

Kvalita „vo veľkom“ a „v malom“.
Návrh dobrého softvéru: zo všetkého najdôležitejšie je riešiť správny problem. Potom vhodne navrhnúť riešenie – integrálnou súčasťou je ho dobre zrealizovať.

Dnes pôjde o zrozumiteľnosť. Autor je len prvý v rade zainteresovaných, nasleduje SQA, údržba ...

Poznámky:

- Aplikácia týchto techník – najlepšie je, keď ju robí SW nástroj.
- “teraz to nejako urobím a potom to upravím (ktovie koľko z toho dovtedy vyhodím)” – nesprávny prístup
- Stručnosť vs. explicitnosť - extrémna stručnosť aj extrémna popisnosť môžu ísť proti zrozumiteľnosti
- tieto pravidlá neubíjajú kreativitu (usmerňujú ju tam, kam patrí)

Výber mien

1. Mená tried a operácií/atribútov – tých, ktorí budú iní používať.

- Používaj plné mená – iba ak by skratky boli dostatočne zaužívané (cpu) – nie takto: Part.Num, Súčiastka.Cis (cislo).
- prípadne aj viacero mien (rocny_prirastok, Testovany_uzol). Triedy môžu mať v prípade potreby aj dlhšie názvy (Absolutny_index_kvality_produktu), pri operáciách/atribútoch radšej nie.
- V názve op/atr neuvádzať názov triedy! (Suciastka.cislo_suciastky)
- Slová v názvoch oddeľovať _, nespájať
- konzistentnosť – priečinok / folder, velkosť / dĺžka
- jazyk – slovenčina / angličtina

2. Lokálne premenné

- tie netreba robiť plnými menami (pred, nasl je častokrát lepšie ako predchodca, nasledovnik) – opäť, kvôli prehľadnosti – príklad na slajd

3. Veľké/malé písmená – dbať na konvencie daného programovacieho jazyka

(C++,MFC –
- triedy a funkcie začínajú veľkými písmenami, zvyšok malé,
- atribúty a premenné malými písmenami,
- konštanty všetko veľké)

4. Gramatické kategórie

triedy: podstatné mená, prípadne kvalifikované
pri štrukturálnom dedení – prídavné mená (comparable – porovnateľný)

rutiny: procedúry, dotazy, atribúty

procedúry: slovesá v neurčitku alebo v rozkazovacom spôsobe (prípadne s prídavkami) – make, move, deposit, set_color: urob, presun, uloz, nastav_farbu

atribúty a dotazy: nie zisti, vrat: (zisti_hodnotu, vrat_hodnotu, get_value – ale hodnota, value).
- Neboolovské atribúty: podstatné meno (prípadne s prídavnými) – číslo, počet, mesačný_súhrn (number, last_month_summary)
- Boolovské: prídavné mená (plný, full), kvôli jednoznačnosti sa v angl. používa „is_“ (is_empty)

Konštanty

“Nepoužívaj hodnotu konštanty (inú ako “nulový prvok”).”

Príklady: 12, “Nemôžem otvoriť súbor”.
new pocty_studentov (1, 12)

Niekedy 1 je konštanta (napr. počet procesorov)

Dá sa zľaviť z reťazcových konštánt používaných práve 1 krát.

Komentáre v hlavičkách

Majú byť! Na úrovni tried (resp. modulov), na úrovni funkcií/atribútov, v kóde.

Rozlíšenie interfejs/implementácia.

Mali by byť stručné. (V prípade “zoznamu funkcií”)

a) nepoužívať “Vráti…” “Vypočíta…” “Táto funkcia…”
b) nepoužívať “the” (angl.)
c) ak jazyk umožňuje pre- a post- conditions, treba ich z komentára vynechať (zbytočná duplicita - “update anomálie”). Ak nemá --- nutne ich uviesť.
d) slovo “bod” je zjavné z deklarácie
e) aktuálna kružnica – nepotrebné

Procedúry – v rozkazovacom spôsobe (“Zaznamenaj nameranú hodnotu.”)
Booleovské dotazy – opytovacia veta (“Nachádza sa *v* v zozname ?”)
V normálnom texte atribúty: ‘v’.
Konzistentnosť – keď je niekde komentár „dĺžka reťazca“, nepoužívať inde „Nastav veľkosť reťazca“.

Používajú sa techniky pre indexovanie (zvláštna klauzula, kde človek píše za týmto účelom)

Kód vnútri procedúr – OK, ale mal by byť o úroveň abstrakcie vyššie ako kód (t.j. nie „zvýši I o 1“) – nie parafrázovať, ale sumarizovať.

Pri nižšieúrovňových jazykoch (assembler, C) treba komentárov viac, pri vyššie stačí menej – navyše, ak je aj vďaka OO prístupu kód modulárny a podelený na menšie, zrozumiteľné, časti.

Text – layout a prezentácia

Je možné spájať viac príkazov na 1 riadok, ak to napomáha prehľadnosti

temp = x; x = y; y = temp;

Odsadenie – dodržiavať konzistený prístup – dobre odsadzovať if, then, else, begin/end, ...
Prázdne riadky v kóde – na oddelenie

Medzery – f(x), f (x) nie f(x)
za čiarkou, nie pred: f (a, b, c)
za bodkočiarkou, nie pred: p1; p2 (x); p3 (y, z) – tolerovateľný je aj francúzsky spôsob – pred bodkoč. (zdôrazňuje symetriu) – ale potom konzistentne
za dvojbodkou, nie pred
pred a za aritmetickými operátormi

Semicolon Style – ak jazyk umožňuje voľbu
- terminatist style – dodržiavať
- inak ich používaj tam, kde je to nevyhnutné

Nested if ... nie hlbšie ako 3 úrovne
príklad – šírka/dĺžka

Výber programovacieho jazyka

- niekedy neprichádza do úvahy (klient si praje, neexistuje, nemáme peniaze)
- X robí COBOL programy 25 rokov
- 200 programátorov, všetci (od jun. po vice-pr.) majú skúsenosti s COBOLom
- Prečo ísť do niečoho iného ako COBOL ?
    - retraining al. najať nových
    - treba udržiavať aj staré programy (napätie medzi COBOL a C++ programátormi)
    - C++ kompilátor a CASE (HW)
    - cenné skúsenosti
- Prečo áno ?
    - ak iná doména – AI-Lisp, komunikácia-C/C++, obrana-Ada
- prototypovanie