Princípy tvorby softvéru
Quality assurance, Testovanie

Robert Lukoťka
lukotka@dcs.fmph.uniba.sk
www.dcs.fmph.uniba.sk/~lukotka

M-255

Quality assurance

Čo znamená kvalita, keď hovoríme o softvéri?

- Funkčná kvalita softvéru
  - ako dobre softvér napĺňa dizajn založený na funkčných požiadavkách alebo špecifikácii.
  - či je vhodný na naplnenie svojho cieľa, prípadne, aký je v porovnaní s konkurenčnými produktami
  - **či bol vytvorený správny softvér**
- Štrukturálna kvalita softvéru
  - ako sú splnené nefunkčné požiadavky viažúce sa k dodaniu funkčných požiadaviek - ako robustnosť, ľahká údržba
  - **či bol softvér vytvorený správne**

Quality assurance

Kvalitný softvér nevznikne sám od seba.

- Kvalitatívne atribúty je potrebné merať.
- Kvalitatívne atribúty je potrebné vynucovať.

Ak chceme získať kvalitný softvér, meranie a vynucovanie kvality musí byť integrálnou súčasťou nášho procesu.

Štrukturálna kvalita softvéru

Diskusia:

- Ako zabezpečiť čitateľnosť kódu / flexibilnosť?

Štrukturálna kvalita softvéru

Diskusia:

- Ako zabezpečiť čitateľnosť kódu / flexibilnosť?
  - code reviews
  - linter (špeciálne dôležite v jazykoch, kde je z dôvodov spätnej kompatibility "bordel") - môže byť súčasťou kompilátora
  - pair programming

Automatické nástroje môžeme integrovať s kompilovaním, VCS, integráciou.

Funkčná kvalita softvéru - Verifikácia a validácia

- Verifikácia - overovanie, či daný artefakt spĺňa požiadavky/špecifikáciu/iné predpísané podmienky
- Validácia - overovanie, či daný artefakt spĺňa zákazníkove potreby

Verifikovať a validovať treba nie len celý produkt, ale aj jednotlivé artefakty.

- Ako validovať požiadavky?
  - nechať zákazníkovi prečítať požiadavky väčšinou nestačí. Číta cca. to čo povedal, je to to čo potrebuje?

Funkčná kvalita softvéru - Verifikácia a validácia

- Verifikácia - overovanie, či daný artefakt spĺňa požiadavky/špecifikáciu/iné predpísané podmienky
- Validácia - overovanie, či daný artefakt spĺňa zákazníkove potreby

Verifikácia

- Manuálne testovanie
- Automatické testovanie

Funkčná kvalita softvéru - Verifikácia a validácia

- Verifikácia - overovanie, či daný artefakt spĺňa požiadavky/špecifikáciu/iné predpísané podmienky
- Validácia - overovanie, či daný artefakt spĺňa zákazníkove potreby

Verifikácia

- Manuálne testovanie
- Automatické testovanie

Kedy preferovať manuálne a kedy automatické testovanie?

Testovanie

Testovanie softvéru je skúmanie softvéru podniknuté na to, aby stakeholderi získali informáciu o kvalite testovaného softvéru alebo služby.

- Štrukturálna a funkčná kvalita majú prienik: komplikovane napísaný kód využívajúci rôzne hacky bude skôr obsahovať zákerný bug.
- Defect detection techniques (Čísla treba brať veeeľmi s nadhľadom, metodika je nejasná).

Testovanie

Testing pyramid

- Testy testujúce menšie celky sú rýchlejšie, môžeme ich mať viac.
- Testov je veľa, automatizácia šetrí veľké množstvo času/zdrojov.
- Pre získanie plných benefitov automatického testovania je potrebné testy tvoriť konzistentne od začiatku projektu.

Unit testy - Závislosti medzi objektami

OO dizajn je založený na spolupráci objektov.

Závislosti:

- creates, destroys
- calls method
- modifies
- ...

Unit test

Unit test

- Solitary - Na najnižšej úrovni separátne testujeme jednotlivé objekty. Všetko ostatné chceme z testu vylúčiť.
- Sociable - Na najnižšej úrovni testujeme objekt spolu s úzko súvisiacimi objektami.

Aj ak preferujeme sociable unit testy, chceme mať kontrolu nad hranicou toho čo je “under test”.

Unit testy - Závislosti medzi objektami

Ako jednotlivé závislosti ovplyvňujú našu schopnosť robiť unit testy?

- creates, destroys
- calls method
- modifies
- ...

Všetky tieto závislosti sú nepríjemné. Tieto závislosti však sú potrebné. Čo s tým?

Dependency injection and dependency inversion

- Objekt by nemal vytvárať inštancie iných objektov, ktoré s ním nie sú úzko previazané (a možno by nemal vytvárať ani takéto inštancie) - dependency injection / dependency injection factory objektu.
- Objekt by nemal závisieť na implementácii kolaborátora, iba na interface - dependency inversion

OO design and object dependence

Ako teda na to?

- creates, destroys - dependency injection (dependency injection factory objektu)
- calls method - interface
- modifies - interface
- ...

Keď chceme testovať, potrebujeme kolaborujúce objekty.

Test doubles

Kolaborujúce objekty sú skryté za interfacami, preto urobíme novú implementáciu týchto objektov - **test doubles**.

M. Fowler: Test doubles

Python unit tests a mocking

- unittest
- mock

Kedy písať testy?

- Po napísaní implementácie.

Kedy písať testy?

- Po napísaní implementácie.
- Počas implementácie
- Medzi designom a implementáciou - výstupy možno použiť pri implementácii.
- Počas designu - garantuje testovateľný dizajn.
- Ako špecifikáciu.
  - Špecifikácia je jednoznačná.
  - Takáto špecifikácia môže byť ťažko čitateľná pre zákazníka.
- Kombinácia postupov.

Písanie testov počas implementácie

- Po naprogramovaní sa poriadne testy chce písať málokomu.
- Jedno z riešení je písať testy priebežne - tri pravidlá TDD (viac harcdore verzia TDD).
  1. You are not allowed to write any production code unless it is to make a failing unit test pass.
  2. You are not allowed to write any more of a unit test than is sufficient to fail; and compilation failures are failures.
  3. You are not allowed to write any more production code than is sufficient to pass the one failing unit test.
- Testy budú pokrývať okrajové prípady.

Ako na GUI?

- Separácia GUI interfacami - testovanie pod GUI.
- Nástroje na automatické GUI testovanie.

Zdroje

- Wikipedia - Software Quality
- Wikipedia - Software Testing