Princípy tvorby softvéru
Quality assurance, Verifikácia, Validácia

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

M-255

Quality assurance

Čo znamená kvalita, keď hovoríme o softveri?

- Software functional quality reflects how well it complies with or conforms to a given design, based on functional requirements or specifications. That attribute can also be described as the fitness for purpose of a piece of software or how it compares to competitors in the marketplace as a worthwhile product. It is the degree to which the correct software was produced.
- Software structural quality refers to how it meets non-functional requirements that support the delivery of the functional requirements, such as robustness or maintainability. It has a lot more to do with the degree to which the software works as needed.

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.

Kvalitatívne parametre, ktoré nesúvisia s požiadavkami

Diskusia:

- Ako zabezpečiť čitateľnosť kódu / flexibilnosť?

Kvalitatívne parametre, ktoré nesúvisia s požiadavkami

Diskusia:

- Ako zabezpečiť čitateľnosť kódu / flexibilnosť?
  - Code reviews.
  - Linter (špeciálne dôležite v jazykoch, kde je z dôvodov spätnej kompatibility "bordel").

Automatické nástroje môžeme indegrovať s kompilovaním, VCS, integráciou (compile suite, commit suite, integration suite).

Verifikácia a validáce

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

Verifikovať a validovať dáva zmysel pre všetky artefakty.

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

Testovanie

Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test.

Defect detection techniques (Čísla treba brať veeeľmi orientačne).

Testovanie

Testing pyramid

- Testy testujúce menšie celky sú rýchlejšie, môžeme ich mať viac.
- Testov je veľa, automatizácia šetrí veľké množstovo 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

Dependency injection and dependency inversion

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

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?

- After writing implementation.

Kedy písať testy?

- After writing implementation.
- After the object is designed but before writing implementation.
  - We may use test during the implementation.

Kedy písať testy?

- After writing implementation.
- After the object is designed but before writing implementation.
  - We may use test during the implementation.
- Before design is complete.
  - Guarantees testable design.
- During specification by example.
  - Specification is unambiguous.
  - We avoid the step specification $\rightarrow$ test cases.
  - Harder to achieve readability for all stakeholders, but many tools emerge.

More

- Tools for to automate GUI testing.
- Separation of GUI.

Zdroje

- Wikipedia - Software Quality
- Wikipedia - Software Testing