Co je peer-review?

Pohled na výsledek činnosti kolegou/kolegyní ze stejného nebo příbuzného oboru.
Primárními cíli jsou:
a) validace kvality výsledku
b) sdílení zkušeností a informací
Speciálním případem peer-review je code-review, kdy dochází ke kontrole zdrojového kódu software.
reviewer - Kvalifikovaná osoba provádějící review.
reviewee - Osoba předkládající výsledek své práce k review.
Pokud není dostupná kvalifikovanější osoba pro provedení review, je vhodné jej provést stejně v rámci předání informací a mechanismu self-review (aneb nutností informace vysvětlit je autor sám validuje).

Kdy se provádí peer-review?

Odevzdáváme hotový výsledek

> ***Reviewee připraví veškeré podklady** pro provedení review, předá je vhodnou formou reviewerovi **a je k dispozici pro případné dotazy**. Reviewer provede review a své poznatky předá/prodiskutuje s **reviewee, který následně zapracuje připomínky**.*

Průběžně během práce

> *Reviewer **průběžně kolaboruje s reviewee** a validuje tak jednotlivé mezivýstupy. Je vhodná **před odevzdáním rychlá validace**. Standardně se používá při technice "párování" (pair--programming).*

Doporučení pro peer-review

Respekt, slušnost a otevřenost je nutný předpoklad pro provádění review.

Review ideálně **provádějte na menším rozsahu a častěji**.

Review **neodkládejte**, ale také **nedělejte předčasně**.

Review **musí být efektivní**, omezte čas na nutné minimum.

Zaměřte se na důležité, nezanořujte se do přílišných argumentací a detailů.

Nekontrolujte, co zkontroluje stroj.

Dohodněte si pravidla pro provádění review ve svém týmu.

Nevymýšlejte kolo a využijte, co již existuje.

Doporučené otázky v rámci review

Jsem spokojený s výsledkem, který odevzdávám k review?

Kdybych byl na jeho místě, co bych udělal jinak? Proč?

Co se může stát a s jakou pravděpodobností, pokud konkrétní nález nebudeme řešit? (řízení rizika)

Code-review checklist

Řešení odpovídá zadání

> *Vyvinutý software odpovídá dohodám, splňuje akceptační kritéria a standardy/praktiky platné v daném kotnextu.*
> *Standardně znamená splnění definovaných akceptačních kritérií, což je vhodná nechat si při předání k review ukázat.*
> *Změna je současně verzována a navázána dle konvencí na zadání (např. identifikace požadavku v commit message ve VCS systému).*

Srozumitelnost a udržovatelnost

> *Kód je pochopitelný bez zbytečné komplexity (simple design), vhodně zdokumentovaný, adekvátně strukturovaný pro splnění základních praktik správného návrhu (KISS, DRY, SOLID, YAGNI).*
> *Kód nevyžaduje dodatečný refaktoring, odpovídá týmovému styleguide.*

Řešení je bez zjevných chyb

> *Kód je zkompilovatelný a provozovatelný (standardně projde CI).*
> *Kód neobsahuje zjevné chyby a známé problematické části.*
> *Standardně používané nástroje nenalezly v kódu relevantní chyby ani upozornění (kompilátor, IDE, lintery a statické analyzátory kódu, aj.).*
> *Kód neosahuje známé antipatterny, vulnerability, riziková místa.*

Defensivní přístup

> *Kontrakty a assumptions v kódu (např. u API) jsou kontrolovány vč. vstupu/výstupu (assertions, kontrolní podmínky).*
> *Výjimky a chyby jsou korektně ošetřeny na vhodné úrovni abstrakce a zdokumentovány jako součást kontraktu API.*
> *Kód je maximálně thread-safe, korektně pracuje se zdroji, nepoužívá zastaralé objekty/operace.*

Otestovanost

> *Kód je pokrytý deskriptivními automatickými testy na vhodné úrovni (unit, contract, integration, e2e), testy slouží jako dokumentace případů užití.*
> *Doporučená sémantika testů je given/when/then. Testy jsou atomické - 1 test pro 1 scénář.*

Code-review checklist (cont)

Telemetrie - logování a monitoring

> *Kód je pokrytý logováním dle principů a standardů logování pro případnou detekci problému (využíváme standardní logovací mechanismy a stejné konvence pro veškerý kód alespoň na úrovni aplikace). Logování neobsahuje nezabezpečené citlivé údaje.*
> *Aplikace je pokryta monitoringem dle principů a standardů monitoringu pro zajištění kontroly za provozu. Aplikace poskytuje telemetrická data pro sledování nefunkčních požadavků a prevenci problémů (např. počet requestů/min, max. a avg. délka odpovědi na request, aj.). Konvence a mechanismy je vhodné stanovit společně s provozovateli aplikace (např. service ownery).*

Jak technicky řešit code review?

Téměř všechny VCS dnes podporují **větve a pull-requesty**, jejichž kombinací je možno technicky nástrojem řešit provádění code review.
Základem je izolace prováděné změny do samostatné větve oddělené od release větví aplikace, její jasná identifikace a po implementaci změny vytvoření pull-requestu jako žádost o review a o začlenění Vaší změny do aplikace. Standardizace tohoto postupu je závislá na zvoleném **VCS workflow**, které je vhodné v týmu definovat a dodržovat.
Využití technického řešení s sebou nese výhody jako evidence historie, podpora online remotingu, vyšší automatizovanost např. napojením výsledků analyzátorů, automatizaci.
Návod pro BitBucket, Github a různá worflows..

Tento cheatsheet přinesla...

Pojďte s námi budovat moderní IT kulturu

Více informací o našem IT naleznete na našich nových kariérních stránkách. Budeme se na Vás těšit.