

# Přednáška 7

## Kvalita softwarových systémů

CORE013 Vývoj softwarových systémů: od myšlenky k funkčnímu řešení

# 7. Kvalita softwarových systémů

- Co je to kvalita, jak ji definujeme
- Jak předcházet problémům s kvalitou
- Jak kontrolovat úroveň kvality
- Testování a jeho role ve vývoji SW

## Domácí práce a příprava na dnešní přednášku

- Procvičení (volitelné) Tableau Public <https://public.tableau.com/en-us/s/download>  
s využitím tutoriálů <https://public.tableau.com/en-us/s/resources>
- Článek [Do remote teams deliver lower quality software?](#)

# CO JE TO KVALITA SW A JAK JI ZAJISTIT?

# Řízení kvality (QA, QC, QE)

- Procesní přístup k udržení požadované úrovně kvality služby nebo produktu s důrazem na každou fázi procesu návrhu, vývoje a dodání.
- Způsob, jak předcházet chybám a vadám výrobků a vyhnout se problémům při dodávání výrobků nebo služeb zákazníkům.
- **Quality Assurance (QA)** a **Quality Control (QC)** se často používá zaměnitelně, ale QA se vztahuje k prevenci vad, zatímco QC se zaměřuje na identifikaci vad.
- **Quality Engineering (QE)** zahrnuje obojí.



# Metody řízení kvality (QE)

Osvědčené postupy  
kódování  
Kódové konvence  
Párové  
programování

Vývoj řízený testy  
Procesy zajištění kvality  
Normy

Funkční testování  
Testování výkonnosti

Testování použitelnosti  
Testování zabezpečení

Měření a metriky  
CMMI, ITIL

Inspekce designu  
Recenze kódu

Statická analýza kódu

Sledování problémů  
Správa konfigurace

Návrhové vzory

Mechanismy odolnosti  
proti chybám  
Ladění výkonnosti

Inženýrství požadavků  
Atributy kvality

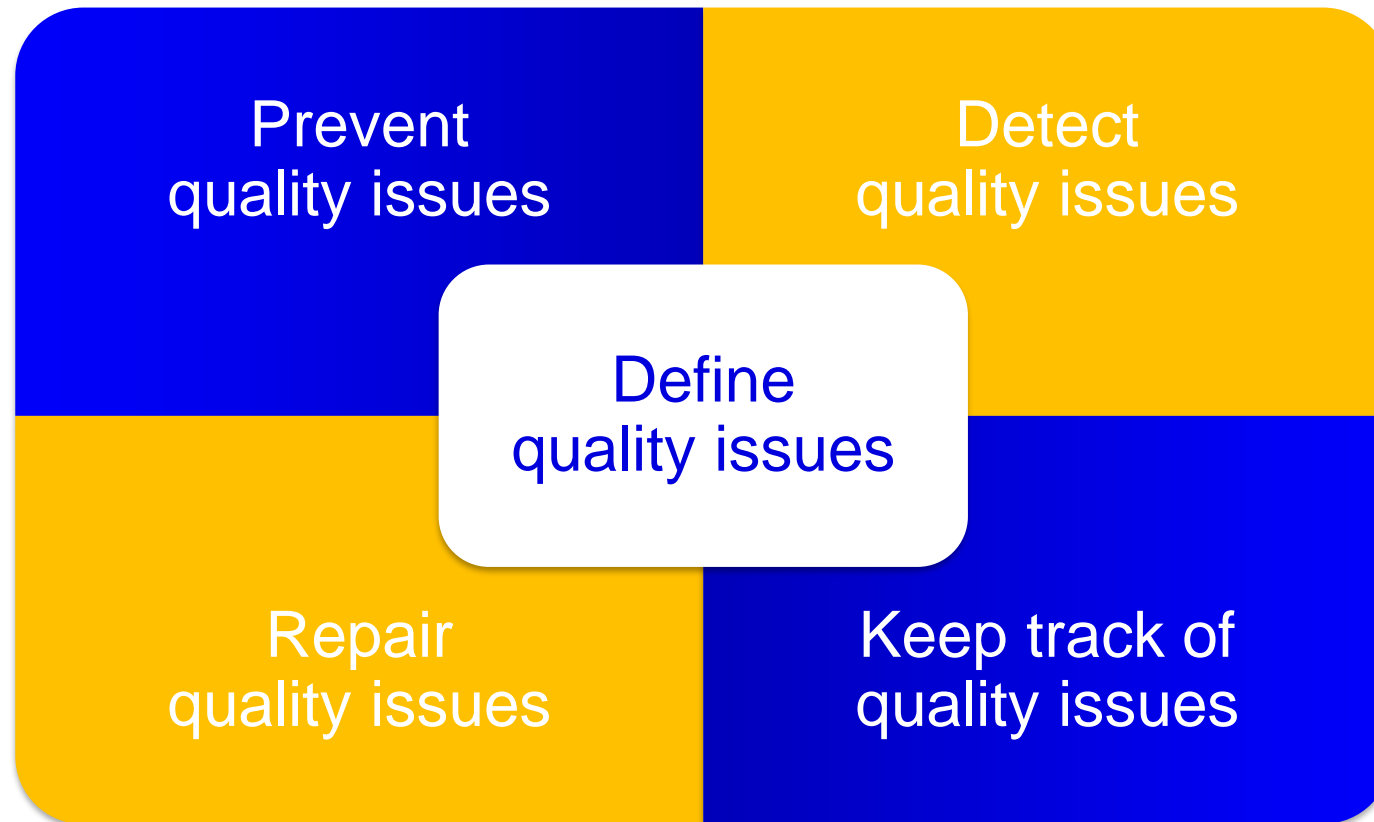
Zásady SOLID  
Čistý kód

V-model testování

Správa technického dluhu

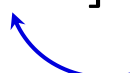
Bezpečnostní taktiky

# Přehled metod QE



# 1. DEFINE

# 1. DEFINE: Definice otázek kvality

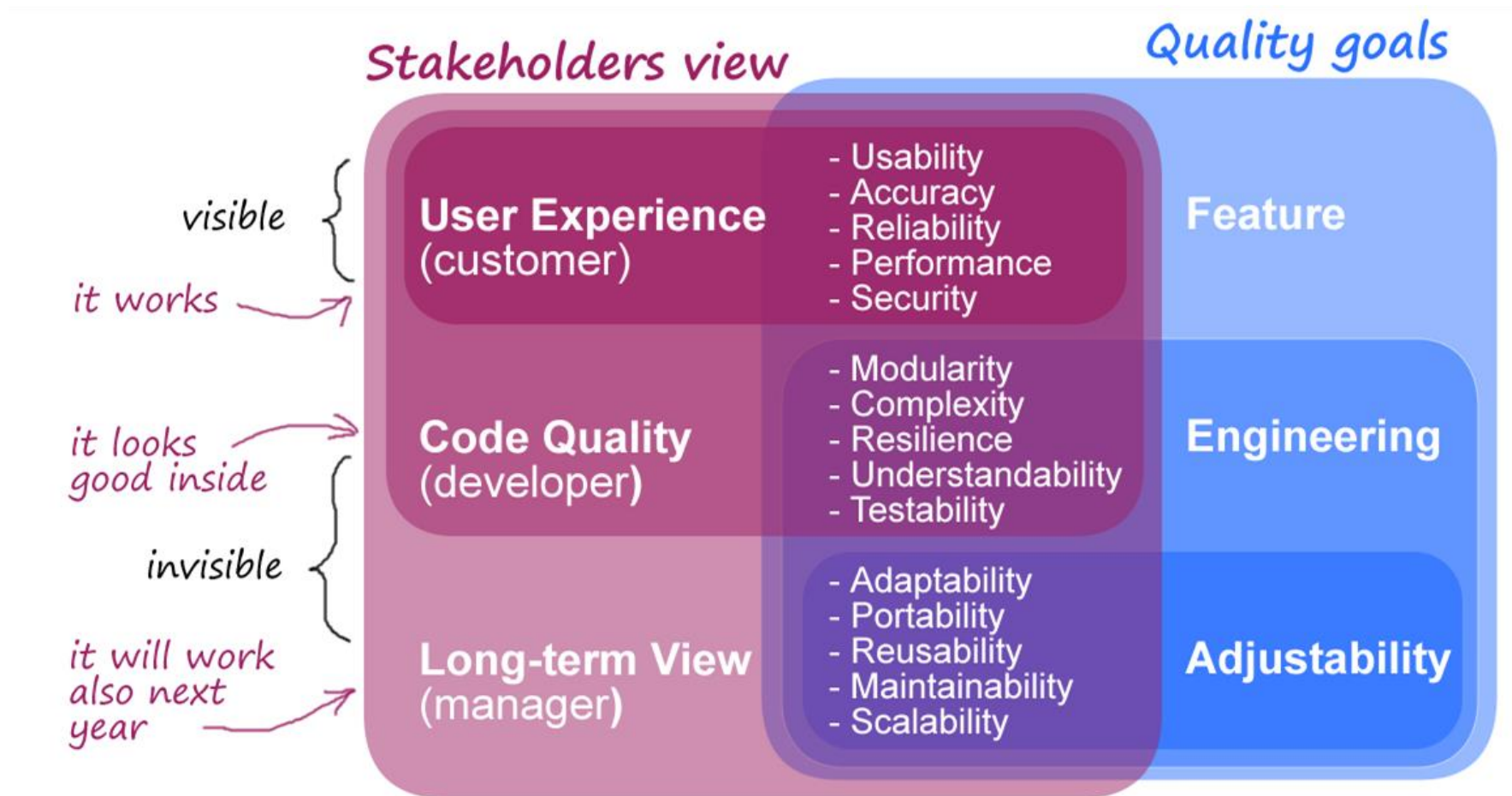
- Kvalita softwaru je běžně definována jako schopnost softwarového produktu přizpůsobovat se požadavkům [ISO/IEC 9001].  
 **potřeby  
zákazníků**
- Specifikace požadavků
- Metriky softwaru
  - "Nemůžete řídit/zlepšovat něco, co neměříte"
- Atributy kvality
  - produktu, procesu a zdrojů



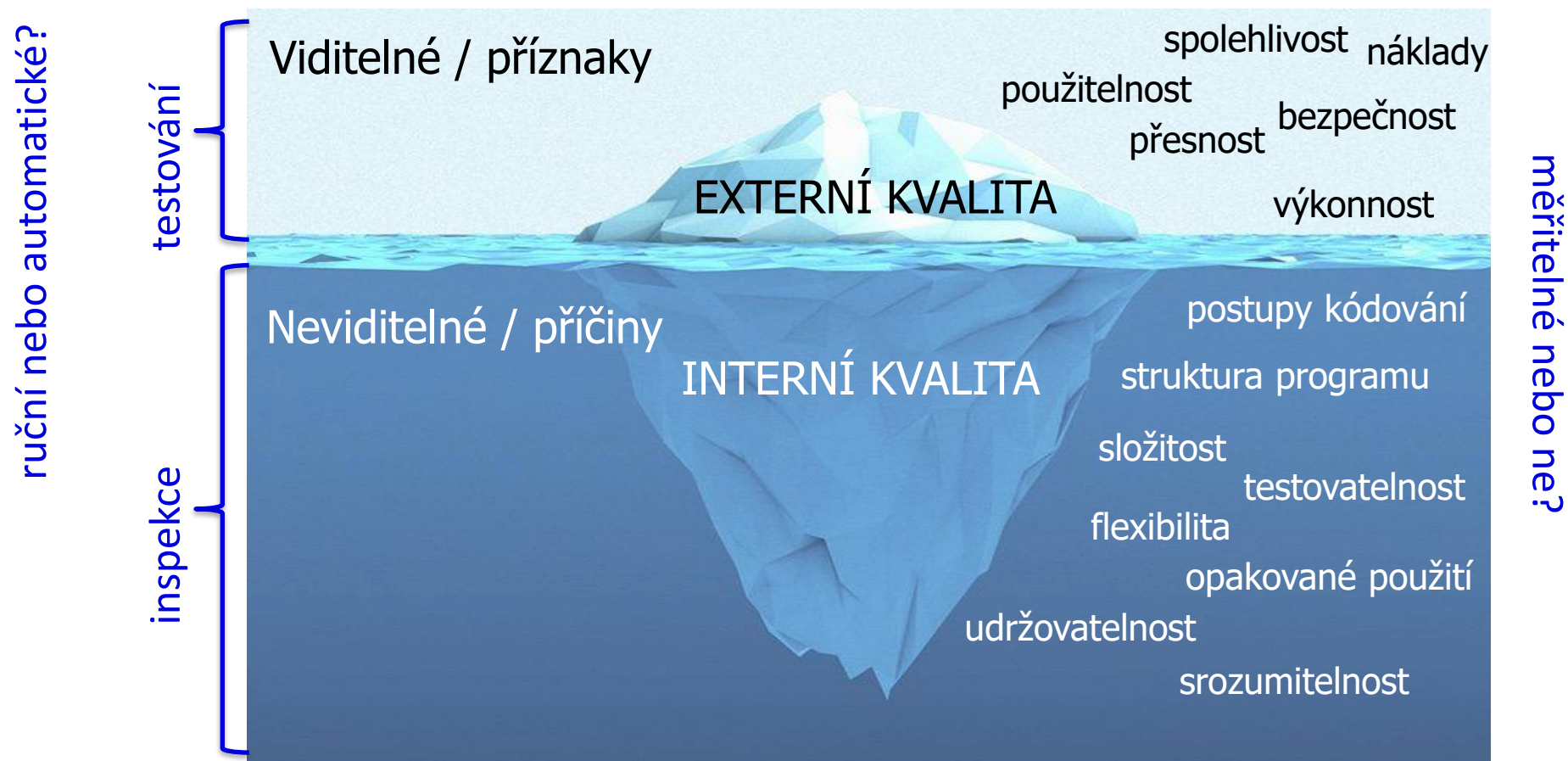


... a pro vašeho zákazníka?

# 1. DEFINE: Co pro vás znamená "kvalita"? ... a vašeho manažera?



# 1. DEFINE: Ledovec kvality softwaru



# 1. DEFINE: Velká pětka

- Udržovatelnost (maintainability)
  - snadná změna (bez zvýšeného technického dluhu)
- Výkonnost (performance)
  - doba odezvy a efektivita využití zdrojů
- Spolehlivost (reliability)
  - pravděpodobnost **bezporuchového provozu** po určitou dobu.
- Bezpečnost (security)
  - schopnost systému **chránit se** před útoky.
- Použitelnost (usability)
  - snadné **používání** systému a **možnost naučit se** jej používat

# 2. PREVENT

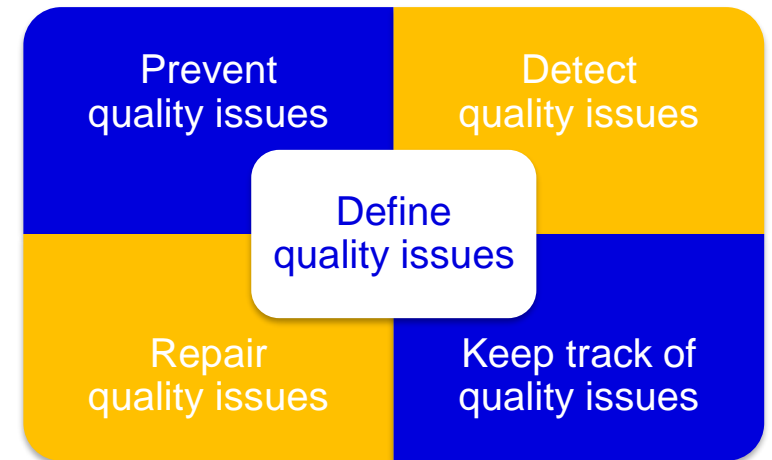
## 2. PREVENT: Předcházení problémům s kvalitou

programování

- Osvědčené postupy kódování
  - Čistý kód, zásady SOLID
  - Návrhové vzory
  - Párové programování
- Konvence psaní kódu
  - Specifická jazyková doporučení

procesy

- Procesy zajišťování kvality
  - V-model testování, vývoj řízený testy
- Standardy pro zlepšování vývojových procesů
  - Referenční modely CMMI a ITIL
  - ISO 9000, ISO/IEC 25010



# 3. DETECT

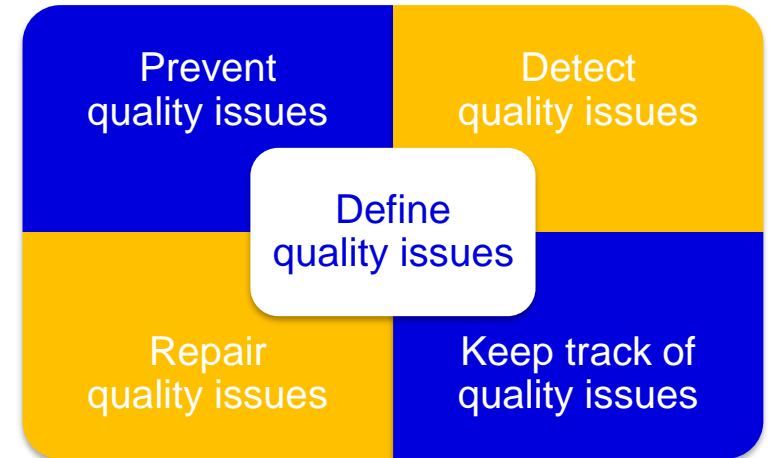
# 3. DETECT: Zjištění problémů s kvalitou

testování

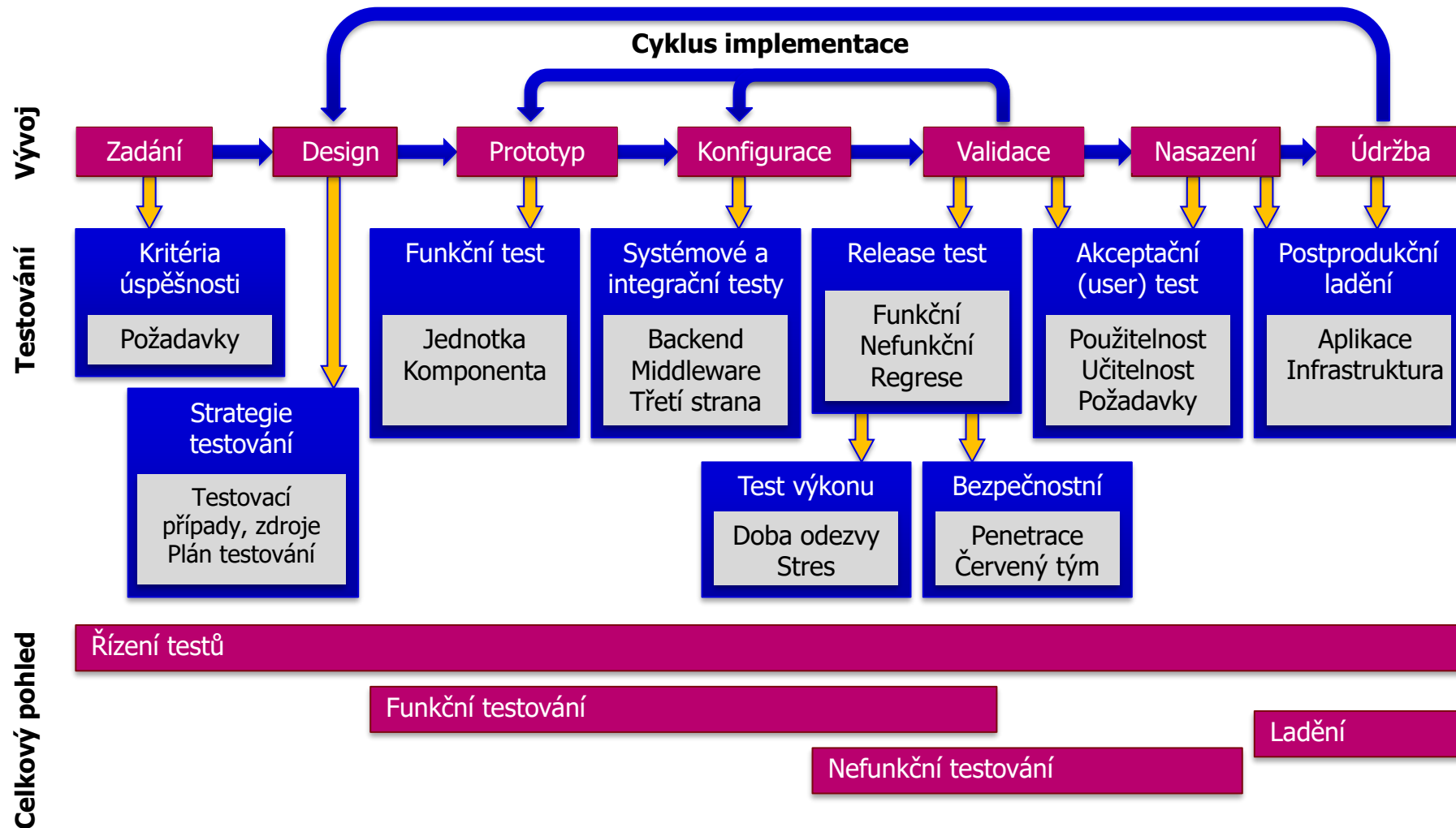
- Testování funkčních požadavků
  - Ruční nebo automatické
- Testování nefunkčních požadavků
  - Výkonnost, použitelnost, zabezpečení testování

statická analýza

- Inspekce designu
  - Ruční kontroly artefaktů návrhu
- Recenze kódu
  - Ruční kontroly kódu
- Automatizovaná statická analýza kódu



# 3. DETECT: Týká se všech fází vývoje – více níže





# Validace vs. verifikace

- **Validace:**

  - "Vytváříme **správný produkt**"

  - Software by měl splňovat očekávání uživatelů, odpovídat jejich požadavkům (specifikace uživatele na vysoké úrovni).

- **Verifikace:**

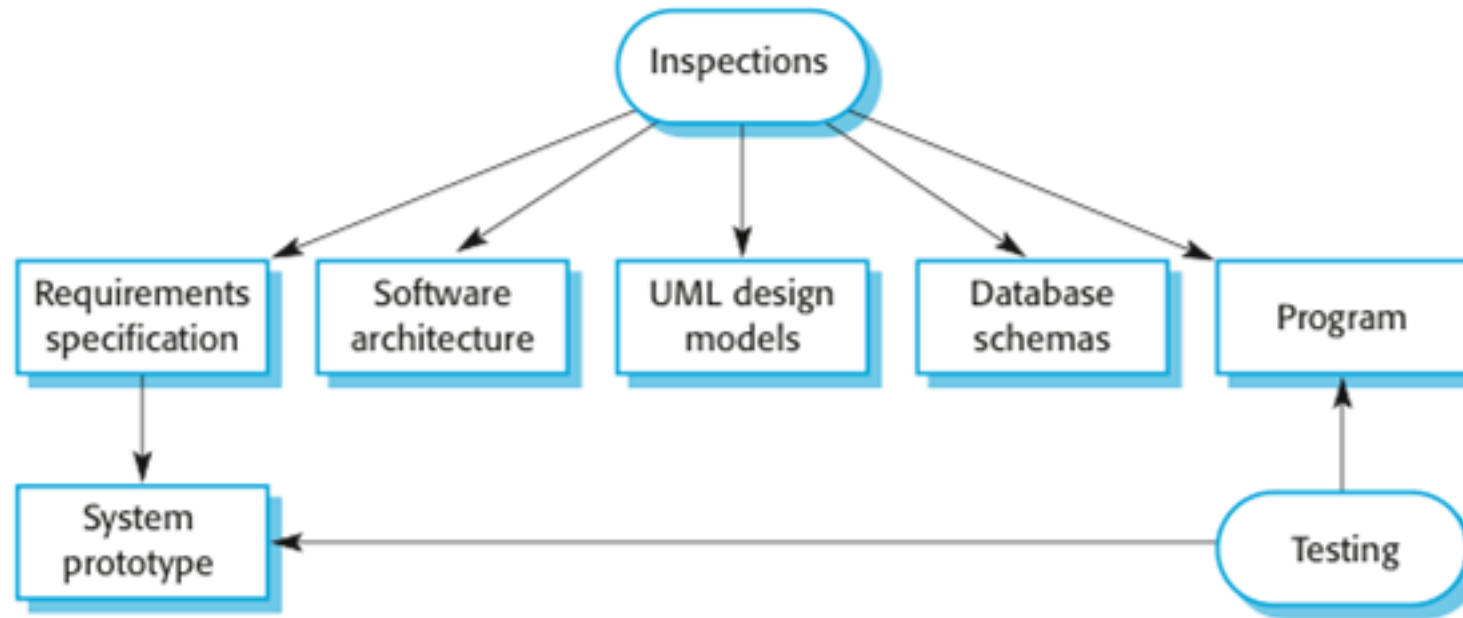
  - "Vytváříme **produkt správně**"

  - Software by měl být bezchybný, měl by odpovídat specifikaci programátora na nízké úrovni (např. testovací případy).

# Dynamické a statické ověřování

- **Dynamické ověřování (tj. testování)**
  - Zabývá se cvičením a pozorováním chování výrobku
  - např. testování softwaru
  - Systém je spuštěn s testovacími daty a je pozorováno jeho provozní chování.
- **Statické ověření (tj. statická analýza)**
  - Zabývá se analýzou statické reprezentace systému s cílem odhalit problémy
  - např. kontroly softwaru
  - Může být doplněn o analýzu dokumentů a kódu pomocí nástrojů.

# Dynamické (testování) a statické (inspekce) metody ověřování



# DYNAMICKÉ METODY DETEKCE (TESTOVÁNÍ)

# Testování programu

- Testování má ukázat, že **program dělá to, co má**, a **odhalit vady programu** před jeho uvedením do provozu.
- Při testování softwaru spouštíte program pomocí umělých dat.
- Zkontrolujete, zda výsledky testování neobsahují chyby a odchylky
- Může odhalit přítomnost chyb, **NE jejich nepřítomnost**.
- Testování je součástí obecnějšího procesu Validace & Verifikace, který zahrnuje také statické techniky V&V.

# Cíle procesu testování

## – Validační testování

- Prokázat vývojáři a zákazníkovi systému, že software splňuje jeho požadavky.
- To znamená, že pro každý požadavek nebo vlastnost systému by měl existovat alespoň jeden test.
- Úspěšný test ukazuje, že systém funguje, jak má.

## – Verifikační testování

- Odhalit chyby nebo závady v softwaru, pokud je jeho chování nesprávné nebo není v souladu s jeho specifikací.
- Úspěšný test je takový test, který způsobí, že systém funguje nesprávně, a odhalí tak chybu v systému.

# Fáze testování

- **Development testing**, kdy se systém testuje během vývoje, aby se odhalily chyby a nedostatky.
- **Release testing**, kdy samostatný testovací tým otestuje kompletní verzi systému před jeho uvolněním pro uživatele.
- **Uživatelské testování**, kdy uživatelé nebo potenciální uživatelé systému testují systém ve svém vlastním prostředí.

# Development testing

- Všechny testovací činnosti, které provádí tým vyvíjející systém.
- **Unit testing/testování jednotek**
  - Testují se jednotlivé programové jednotky nebo třídy objektů. Unit testing by se mělo zaměřit na testování funkčnosti objektů nebo metod.
- **Systémové/integrační testování**
  - některé nebo všechny součásti systému jsou integrovány a systém je testován jako celek. Systémové testování by se mělo zaměřit na testování interakcí komponent.



# Release testing

- Proces testování **konkrétní verze systému**, která je určena k použití.
- Hlavním cílem procesu testování při uvolnění je přesvědčit dodavatele systému, že je **připravený pro použití**.
  - Testování při uvedení do provozu proto musí prokázat, že systém poskytuje stanovené funkce, výkonnost a spolehlivost a že při běžném používání neseleže.
  - Případy užití vytvořené za účelem identifikace požadavků na systém lze použít jako základ pro release testing.
- Testování verze je obvykle **black-box procesem**, kdy jsou testy odvozeny pouze ze specifikace systému.

# Uživatelské testování

- Fáze procesu testování, ve které **uživatelé nebo zákazníci** poskytují vstupní informace a rady k testování systému.
- Uživatelské testování je nezbytné i v případě, že bylo provedeno komplexní testování systému a release testy.
  - Důvodem je skutečnost, že vlivy pracovního prostředí uživatele mají zásadní vliv na spolehlivost, výkonnost, použitelnost a robustnost systému.

# STATICKÉ METODY DETEKCE (STATICKÁ ANALÝZA)

# Statická analýza

- Techniky statické analýzy jsou techniky ověřování systému, které **nezahrnují spuštění programu**.
- Statická analýza zahrnuje techniky, jako jsou
  - Kontroly a code reviews - **manuální**
  - Automatizovaná analýza programu - **automatické**
- Statická analýza má svůj význam vždy, když je **levnější najít a odstranit chyby než platit za selhání systému** – např. v kritických systémech.

# Výhody statických metod – inspekcí

- Během testování **mohou chyby maskovat jiné chyby**. Protože kontrola je statický proces, nemusíte se zabývat interakcemi mezi chybami.
- **Neúplné verze** systému lze kontrolovat bez dalších nákladů. Pokud je program neúplný, je třeba vyvinout specializované testovací makety pro testování částí, které jsou k dispozici.
- Kromě hledání chyb programu může kontrola zohlednit i další atributy kvality programu, jako je **dodržování standardů, přenositelnost a udržitelnost**.

# Automatizovaná statická analýza – SONARQUBE

```
246 if (Provider.class == roleTypeClass) {  
247     Type providedType = ReflectionUtils.getLastTypeGenericArgument(dependencyD  
248     2 Class providedClass = 1 ReflectionUtils.getTypeClass(providedType);  
249  
250     if (this.componentManager.hasComponent(providedType, dependencyDescriptor.  
251         || 3 providedClass.isAssignableFrom(List.class) || providedClass.isA  
  
252     continue;  
253 }
```

A "NullPointerException" could be thrown; "providedClass" is nullable here.

 Bug

 Major

 cert, cwe

## RELIABILITY

 0  A Bugs

Quality Gate  
**Passed**

All conditions passed

## SECURITY

 0  A Vulnerabilities

 1 Hotspots

## MAINTAINABILITY

 4 Code Smells

 5  A Debt  
min

# DETEKCE NEDODRŽENÍ NEFUNKČNÍCH ATRIBUTŮ KVALITY

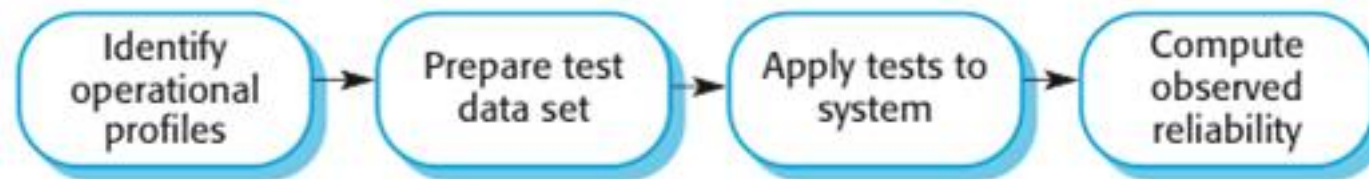
# Testování výkonnosti

- **Testování verze** může zahrnovat testování nových vlastností systému, jako je výkonnost a spolehlivost.
- Testy by měly odrážet **profil použití** systému.
- Testy výkonnosti obvykle zahrnují naplánování série testů, při nichž se **zátěž postupně zvyšuje**, dokud se výkonnost systému nestane nepřijatelnou.
- **Zátěžové testování** je forma testování výkonnosti, kdy je systém záměrně přetížen, aby se otestovalo jeho chování při selhání.



# Ověřování spolehlivosti

- Ověřování spolehlivosti zahrnuje cvičení programu, jehož cílem je posoudit, zda program dosáhl požadované úrovně spolehlivosti.
- Měření spolehlivosti vyžaduje speciálně navržený soubor dat, který odráží **očekávané použití systému** (na rozdíl od testování defektů, kde je atypickému použití věnována větší pozornost), tj. **kopíruje očekávané vstupní vzorce**.



# Ověřování bezpečnosti

- **Testování**, do jaké míry se systém dokáže chránit před vnějšími útoky.
- Problémy s testováním bezpečnosti
  - Bezpečnostní požadavky jsou **požadavky typu "nesmí se"**, tj. určují, co by se nemělo stát. Obvykle není možné definovat bezpečnostní požadavky jako jednoduchá omezení, která lze kontrolovat systémem.
  - **Lidé, kteří útočí na systém, jsou inteligentní** a hledají zranitelná místa. Mohou experimentovat a objevovat slabiny a mezery v systému.
- **Statická analýza** může být použita k navedení testovacího týmu na oblasti programu, které mohou obsahovat chyby a zranitelnosti.

# Ověřování bezpečnosti

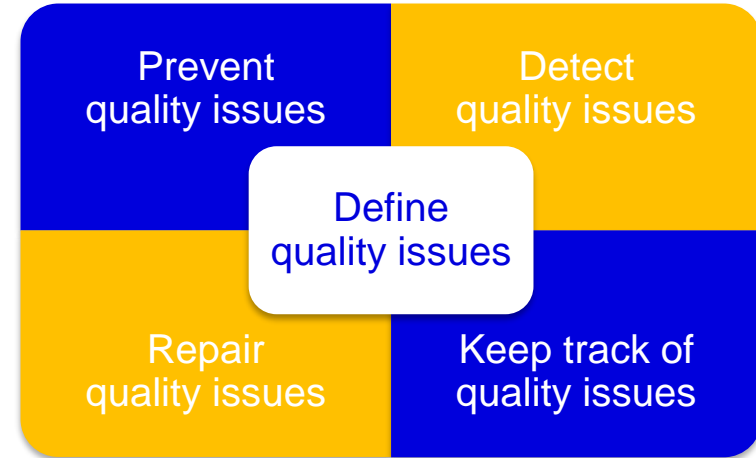
- **Ověřování na základě zkušeností**
  - Systém je analyzován z hlediska typů útoků, které jsou validačnímu týmu známy.
- **Tiger teams**
  - Je vytvořen tým, jehož cílem je prolomit zabezpečení systému simulací útoků na systém.
- **Validace pomocí nástrojů**
  - K analýze provozovaného systému se používají různé bezpečnostní nástroje, jako jsou například nástroje pro kontrolu hesel.
- **Formální verifikace**
  - Systém je ověřen podle formální bezpečnostní specifikace.

# 4. REPAIR

## 4. REPAIR: Náprava kvality

- Funkční problém
  - Oprava kódu
- Problém se spolehlivostí
  - Mechanismy odolnosti proti chybám
- Problém s výkonností
  - Souběžnost, efektivní využití zdrojů, identifikace a odstranění úzkých míst systému
- Problém s bezpečností
  - Identifikace a odstranění zranitelných míst systému (single points of failure).
- Problém s udržitelností
  - Refaktoring podle zásad čistého kódu, návrhových vzorů

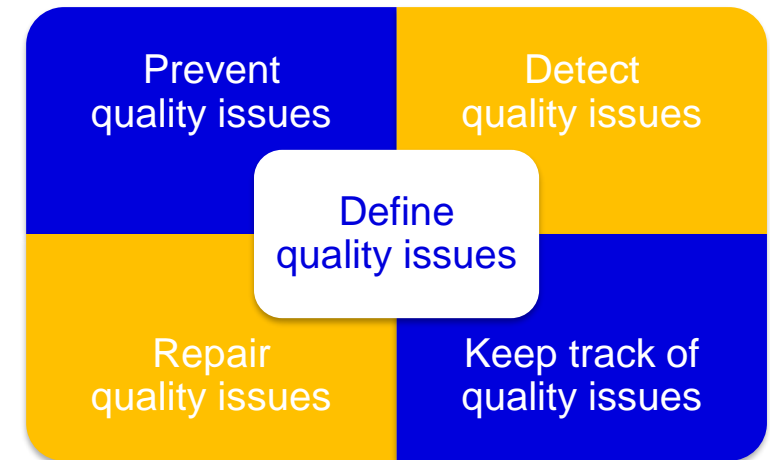
taktiky a vzory



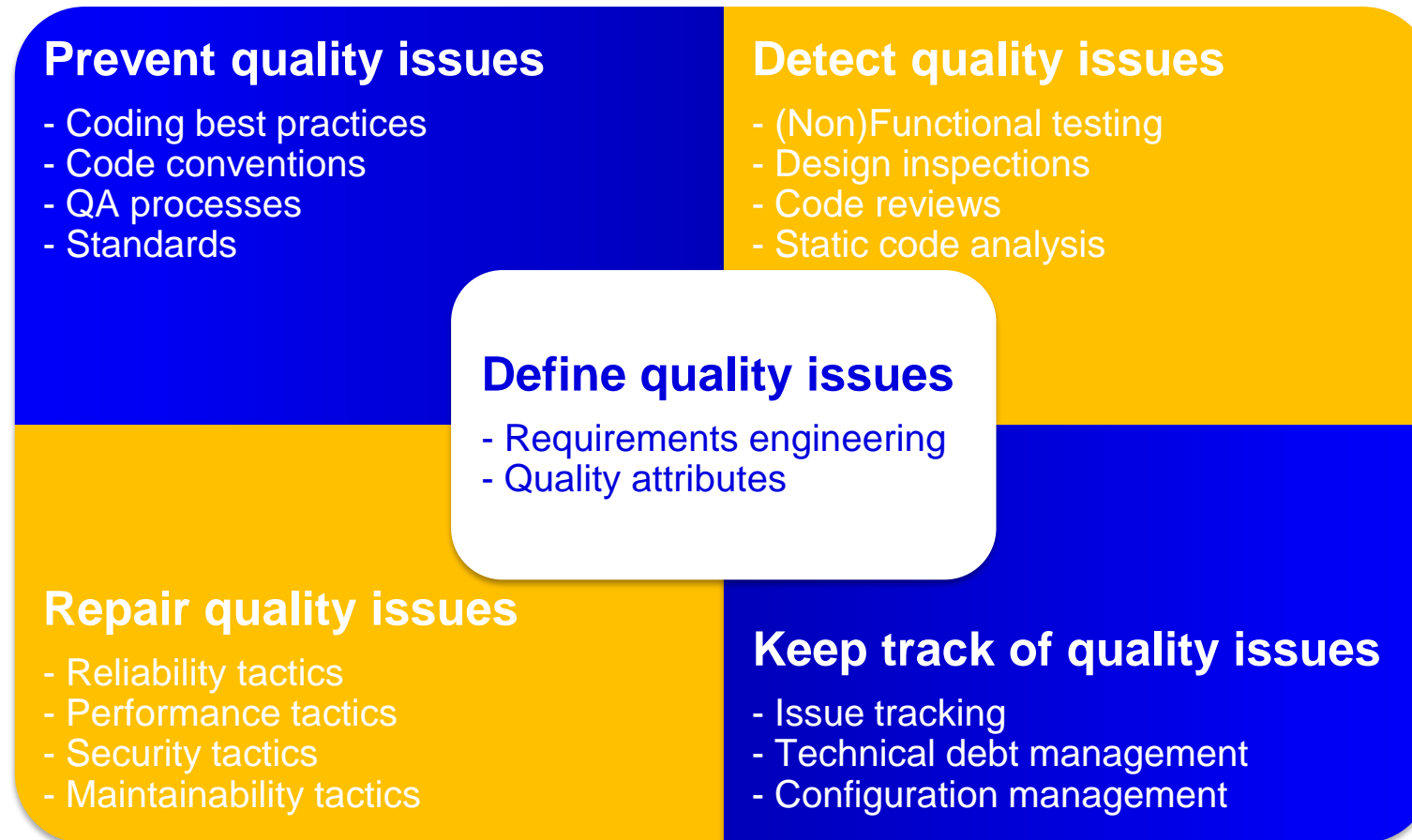
# 5. TRACK

# 5. TRACK: Sledování vývoje kvality

- Sledování problémů
  - Podporuje správu problémy nahlášené zákazníky
- Správa technického dluhu
  - Úroveň zhoršení kvality kódu
  - Práce, kterou je třeba provést před tím, než může být určitý úkol považován za dokončený
- Správa konfigurace
  - Správa verzí a správa vydání
  - Integrace systému



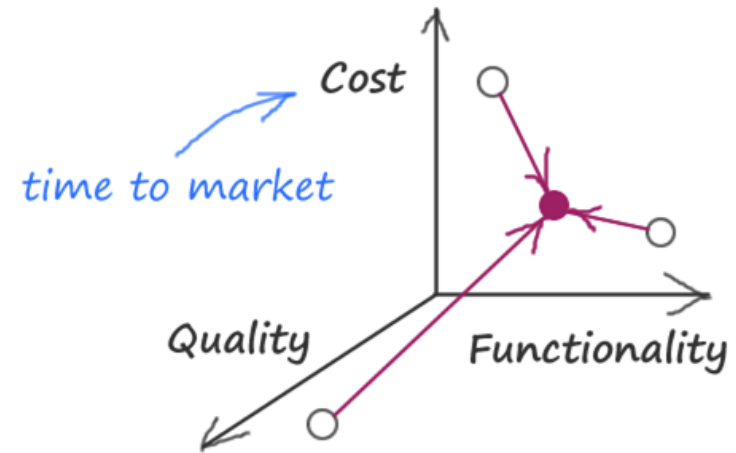
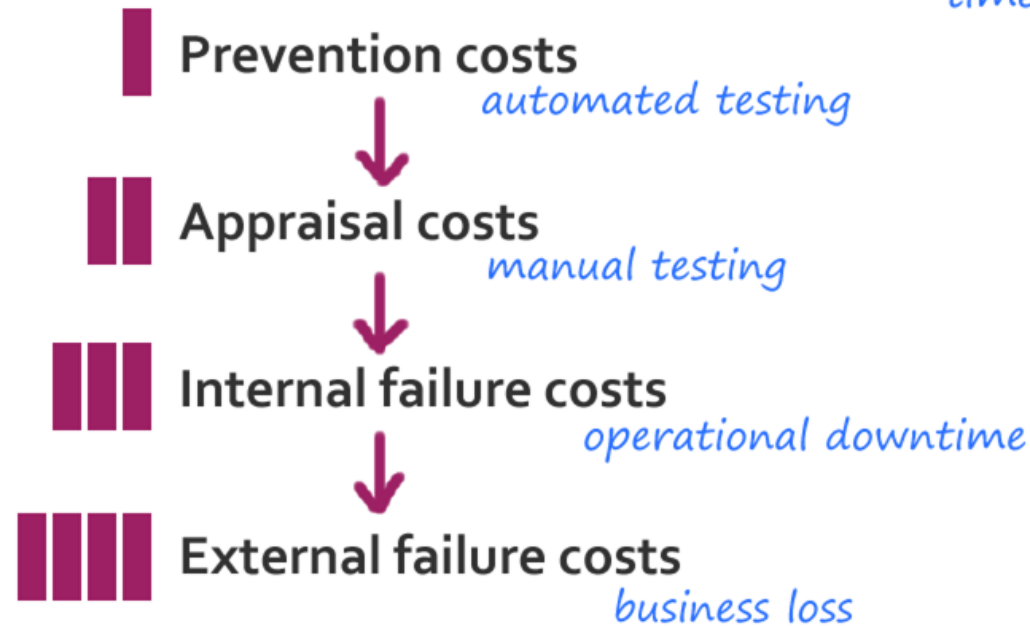
# Přehled metod QE





# Dobře si vybrat, dobře naplánovat

- Dobře si promyslete své požadavky a náklady na kvalitu



# Dobře si vyberte - klíčová je kombinace



# Dobré plánování - Síla analogie

## – Servis letadel

- Vyžaduje pravidelný servis, např. každých 100 000 km.
- Probíhá, i když se zdá, že vše funguje v pořádku, protože si nemůžeme dovolit selhání.

## – Správa technického dluhu

- Představil Ward Cunningham
- Analogie zhoršení kvality s finančním dluhem - pokud není splacen, úroky se zvyšují. Člověk se může dostat do problémů.

**Můžeme to kvantifikovat?**



## – Někdy je rozumné si "půjčit"

- Když člověk očekává, že bude mít v budoucnu více prostředků než nyní (začínající firma).
- Když je třeba jednat rychle, abychom nepropásli tržní příležitost
- Kdy se očekává znehodnocení prostředků (např. vývojáři budou zkušenější, bude snazší pochopit potřeby uživatelů).

# Shrnutí

- **Quality Engineering (QE)** je mnohem víc než jen **testování**, které zahrnuje mnoho různých metod, jak
  - prevence, odhalování, opravy a sledování problémů s kvalitou.
- Klíčem k úspěšnému QE je **kombinace metod**.
  - Dobře si však vyberte a naplánujte, ne všechny metody jsou pro váš projekt nejlepší!
- Ujistěte se, že rozumíte **potřebám svých zákazníků**
  - Vyvážit interní i externí atributy kvality pro obě strany. současnost i budoucnost

# Odkazy

- [1] Testování, které provádíte při vývoji aplikace Siebel. Dostupné online na adrese [http://docs.oracle.com/cd/E14004\\_01/books/DevDep/Overview5.html](http://docs.oracle.com/cd/E14004_01/books/DevDep/Overview5.html)
- [2] Steve McConnell. Code Complete: S.: Code Code: Praktická příručka pro tvorbu softwaru, druhé vydání. Microsoft Press, červen 2004.
- [3] Kevin Burke. Why code review beats testing: evidence from decades of programming research [Proč kontrola kódu překonává testování: důkazy z desetiletí výzkumu programování]. Dostupné online na adrese <https://kev.inburke.com/kevin/the-best-ways-to-find-bugs-in-your-code/>
- [4] RebelLabs. Zpráva o produktivitě vývojářů za rok 2013. Dostupné online na adrese <http://zeroturnaround.com/rebellabs/developer-productivity-report-2013-how-engineering-tools-practices-impact-software-quality-delivery/>
- [5] Jonathan Bloom. Titanic Dilemma: Viditelné versus neviditelné. Dostupné online na <http://blog.castsoftware.com/titanic-dilemma-the-seen-versus-the-unseen/>

# CO NÁS ČEKÁ PŘÍŠTĚ

## 8. Klíčové atributy kvality

- Udržovatelnost
- Výkonnost
- Spolehlivost
- Bezpečnost
- Použitelnost

### Domácí práce a příprava na příští přednášku

- Podívejte se na možnosti nástroje [SONARQUBE](#)
- Přijďte se k nám na Den s průmyslovými partnery FI MU

<https://www.fi.muni.cz/for-partners/meeting-autumn2023.html.cs>