Financované Európskou úniou
NextGenerationEU

PLÁN OBNOVY

MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY

KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI
ŽILINSKEJ UNIVERZITY V ŽILINE

Bezpečný vývoj a testovanie softvéru

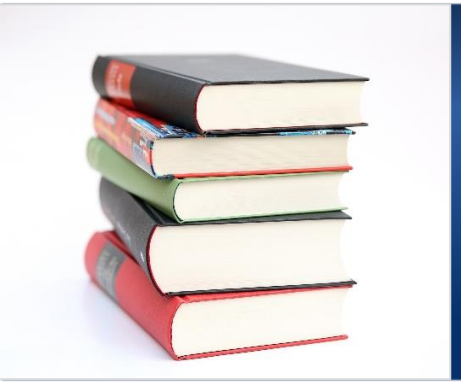
Technické opatrenia (Blok IV)

Kurz: Manažér kybernetickej bezpečnosti

doc. Ing. Jozef Kostolný, PhD.

KC KYB UNIZA, https://kc.uniza.sk

jozef.kostolny@fri.uniza.sk



Obsah

- Metódy a techniky softvérového inžinierstva vrátane modelov vývoja softvéru
- Princípy životného cyklu vývoja systémov a zásady bezpečného vývoja softvéru
- Metódy testovania a vyhodnocovania bezpečnosti systémov

Metódy a techniky softvérového inžinierstva vrátane modelov vývoja softvéru

Softvérového inžinierstvo

- Softvérové inžinierstvo sa zameriava na **systematický, organizovaný a efektívny vývoj softvéru** a využíva pri tom rôzne **metódy** (prístupy) a **techniky** (praktiky).
- **Metódy vývoja softvéru** – ide o prístupy, ktoré určujú, ako sa softvér bude navrhovať, implementovať a spravovať:
  - **Štruktúrované metódy:**
    - Rozdelenie softvéru na moduly (diagramy toku dát).
  - **Objektovo-orientované metódy:**
    - Softvér sa modeluje ako súbor objektov (tried), ktoré komunikujú (UML diagrame).
  - **Agilné metódy:**
    - Dôraz na iteráciu, spätnú väzbu a zmenu požiadaviek.
    - Príklady: Scrum, Kanban, Extreme Programming
  - **Formálne metódy:**
    - Používajú sa matematické modely na verifikáciu správnosti softvéru.

Financované Európskou úniou NextGenerationEU | PLÁN OBNOVY | MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY | KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

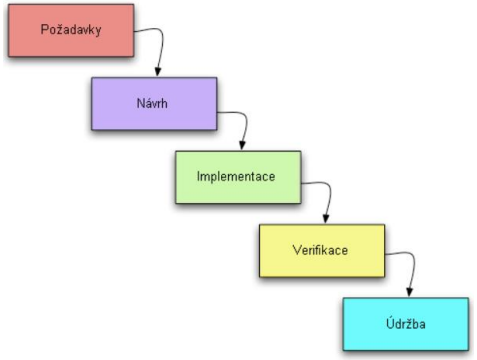
Techniky softvérového inžinierstva

- Ide o praktické nástroje a procesy, ktoré podporujú jednotlivé fázy vývoja.

Fáza SDLC	Techniky
Zber požiadaviek	Interview, dotazníky, analýza dokumentácie, brainstorming
Analýza systému	DFD, ER diagramy, Use Case diagramy
Návrh systému	UML (triedy, sekvenčné diagramy), návrhové vzory (design patterns)
Implementácia	Štandardy kódovania, verzionovanie (Git), CI/CD
Testovanie	Jednotkové testy, integračné testy, SAST, DAST
Údržba a prevádzka	Refaktoring, bug tracking, monitorovanie, patch management

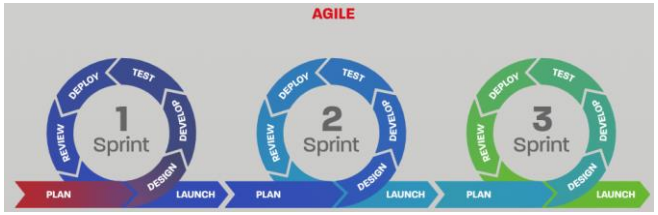
Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

Modely vývoja softvéru

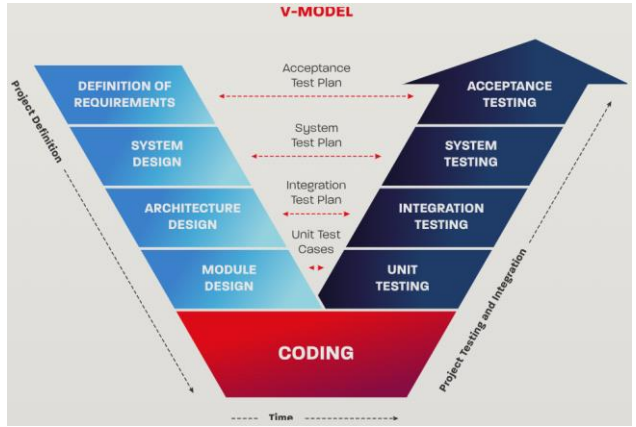


Popisujú sekvenciu a organizáciu aktivít vo vývojovom procese. Medzi najznámejšie patria:

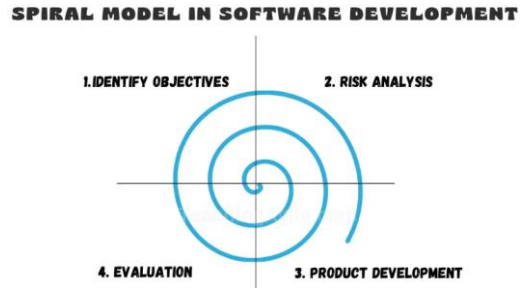
- **Vodopádový model (Waterfall)**
  - Lineárny, sekvenčný prístup: každá fáza sa ukončí pred začiatkom ďalšej.
  - Vhodný pre projekty s jasnými požiadavkami.
  - Nevhodný pre časté zmeny.
- **Iteratívny a inkrementálny model**
  - Vývoj prebieha v cykloch – systém sa rozvíja postupne.
  - Každá verzia zvyšuje funkcionalitu.
- **Agilný model (Agile)**
  - Dôraz na rýchlu dodávku, spoluprácu a flexibilitu.
  - Softvér sa vyvíja v iteráciách (sprintoch).



- **V-model (Validation & Verification)**
  - Rozšírenie vodopádového modelu s dôrazom na testovanie v každej fáze.
  - Každá fáza vývoja má zodpovedajúcu testovaciu fázu.

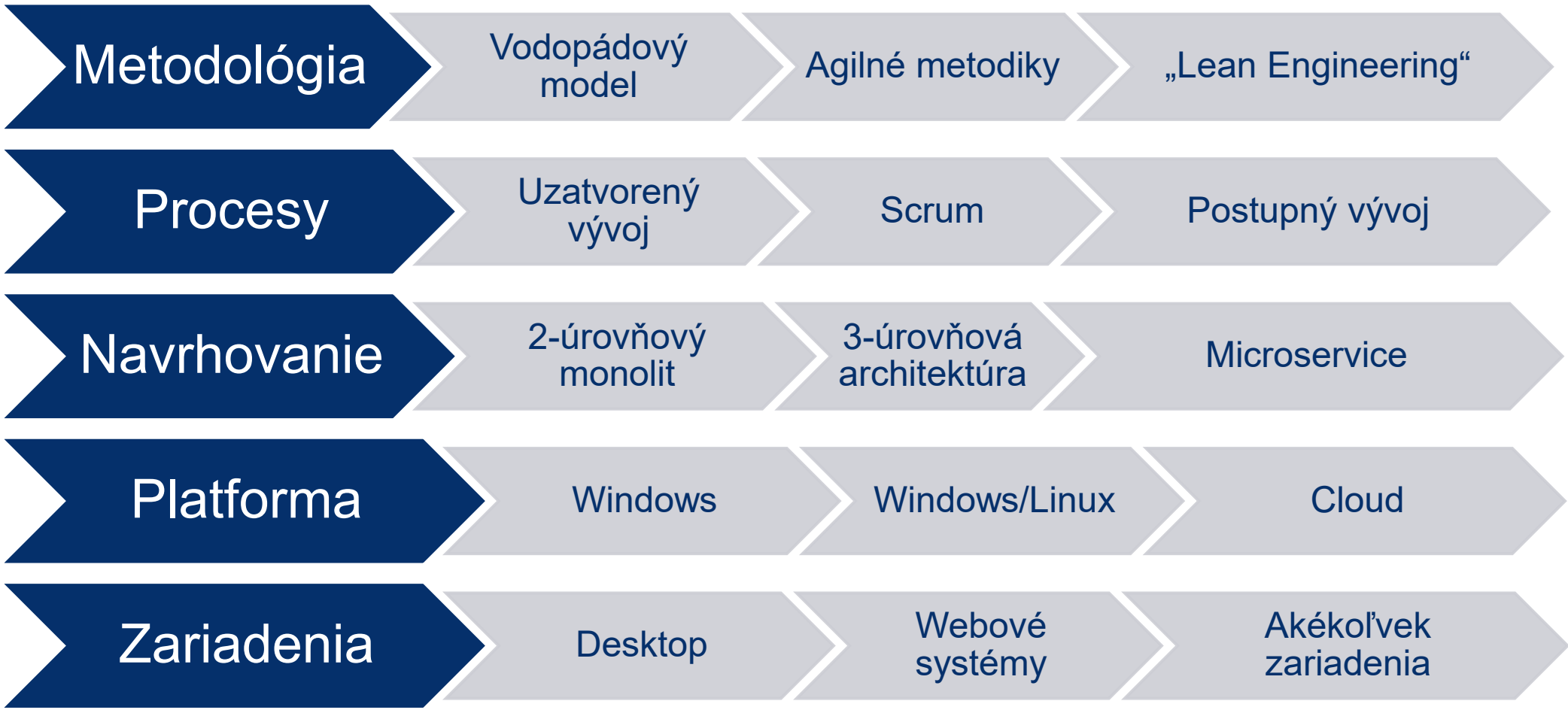


- **Spiral model**
  - Kombinácia iteratívneho vývoja a riadenia rizík.
  - Vývoj prebieha v „špirálach“ – v každej sa analyzujú riziká, návrh, implementácia a testovanie.



- **DevOps model**
  - Prepojenie vývoja (Dev) a prevádzky (Ops).
  - Automatizované testovanie, nasadzovanie a monitorovanie.
  - CI/CD, kontajnery (Docker), orchestrácia (Kubernetes).

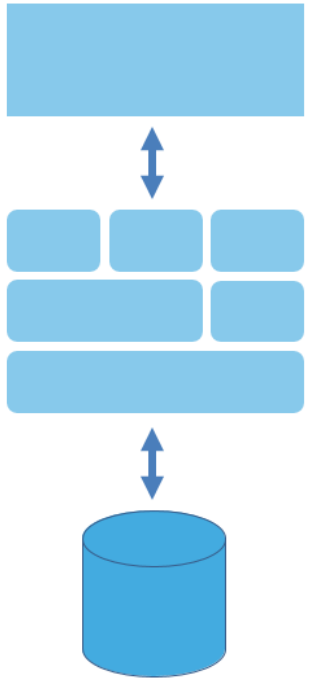
Techniky vývoja softvéru



Techniky vývoja softvéru

Minulosť

- Robustná klient-server aplikácia na robustnom klientovi
- Dobre zadefinované prostredie – OS, využitie, na čom to bude bežať
- Monolitická aplikácia, fyzické médium, infraštruktúra



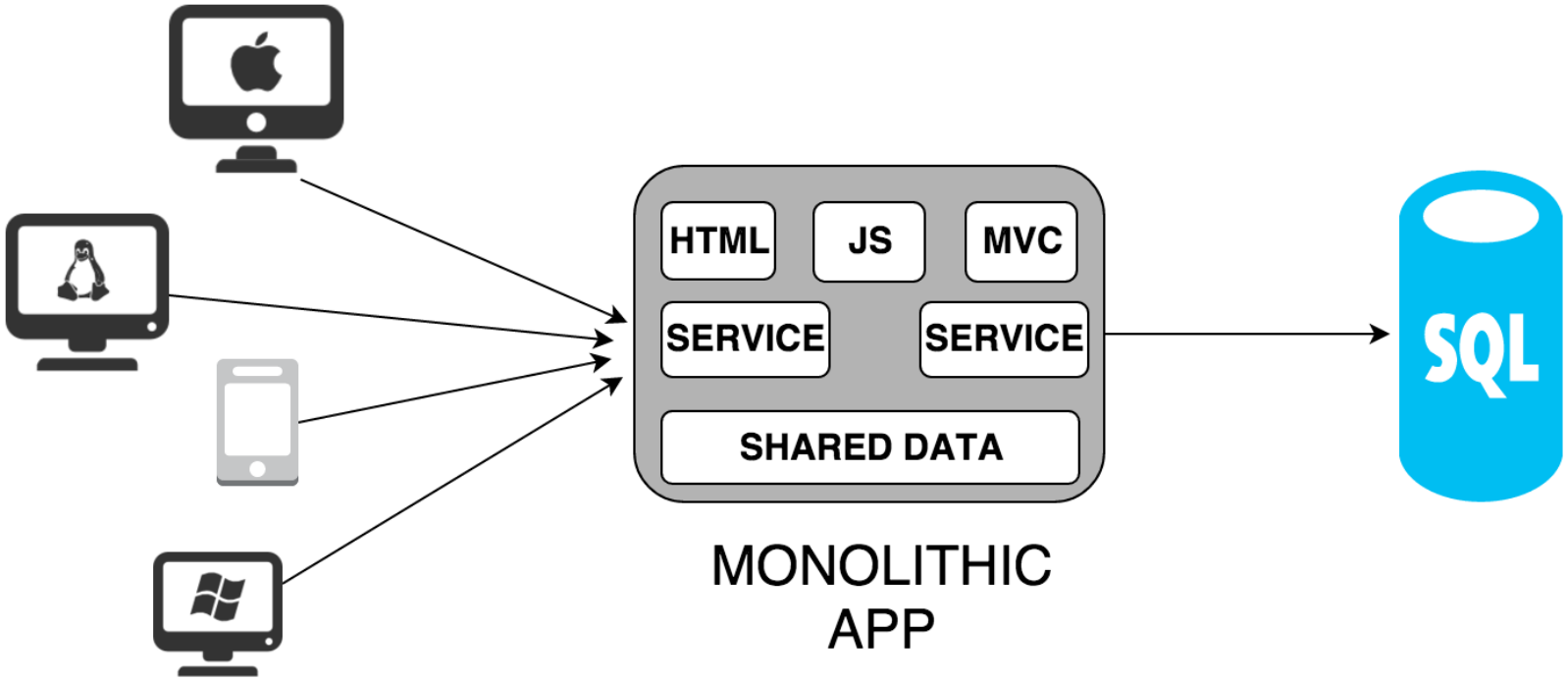
Súčasnosť

- Robustná aplikácia na akomkoľvek zariadení
- Zložená na najlepšie dostupné služby
- Bežiaca na dostupnej zostave fyzických zdrojov

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

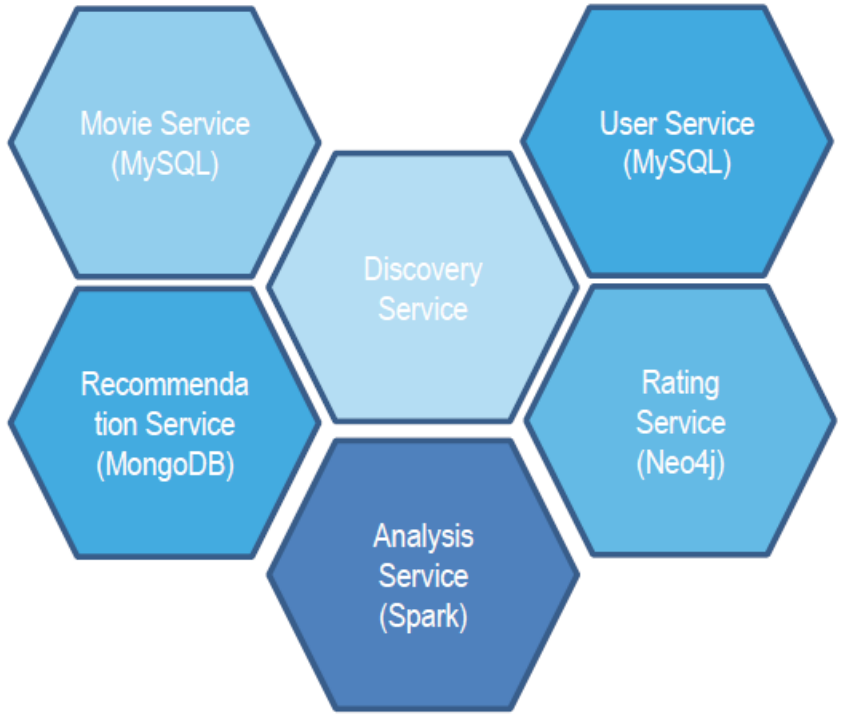
Techniky vývoja softvéru

- **Čo je služba?**



Techniky vývoja softvéru

- **Čo je to Microservice?**
  - Voľne prepojená architektúra orientovaná na služby s ohraničenými obsahom
  - Návrh softvéru ako zostava nezávislých služieb voľne prepojených prostredníctvom jednoduchej komunikácie
    - Aplikácia je rozdelená na malé časti
    - Každá služba je postavená na podnikovej kapacite zodpovedná za seba samú
    - Každá služba beží ako samostatný proces
    - Jednoduché prepojenie vnútorným mechanizmom napr. HTTP
    - Minimum centralizovaného riadenia služieb
    - Každá služba môže byť naprogramovaná rôznym jazykom na rôznej technológii



Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

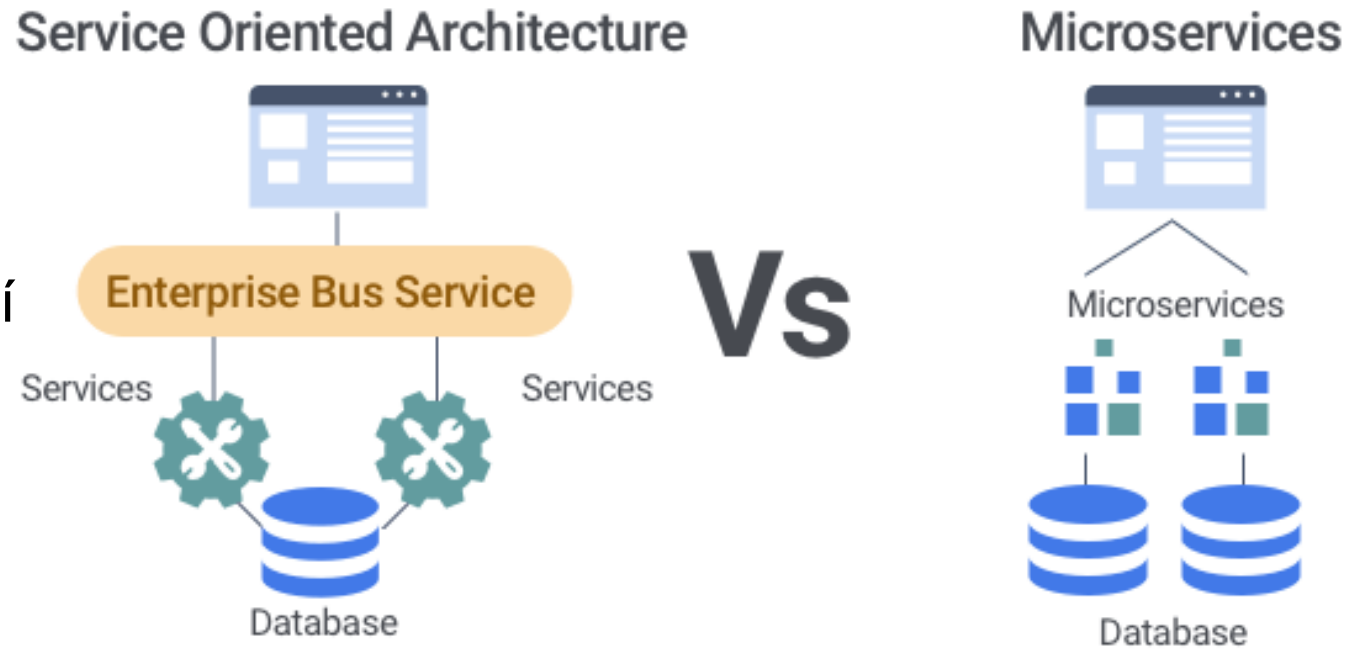
Techniky vývoja softvéru

- **Návrhové princípy microservice**
  - Vysoká súdržnosť systému
    - Zameranie sa na časti, ktoré fungujú správne
    - Rozdeľovanie na jednotlivé služby
  - Autonómne
    - Nezávislé zmeny
    - Nezávislé vydávanie
    - Jednoduchá komunikácia, verzionovanie, vlastníctvo služby vývojárskym tímom
  - Zameranie na podnikovú doménu
    - Prezentovanie podnikových funkcií
    - Možnosť rozdelenia na podskupiny
  - Pružnosť
    - Predpokladať zlyhanie
    - Predurčená funkcionalita zlyhávania
    - Návrh pre známe možnosti zlyhania
    - neočakávané zlyhanie rýchla oprava
  - Monitorovanie
    - Možnosť sledovania stavu systému
    - Centralizované monitorovanie
    - Nástroje pre real-time monitorovanie a logovanie
  - Automatizácia
    - Nástroje pre testovanie a odozvu
    - Nástroje na vydávanie
    - Postupná integrácia nástrojov

Prečo microservice?

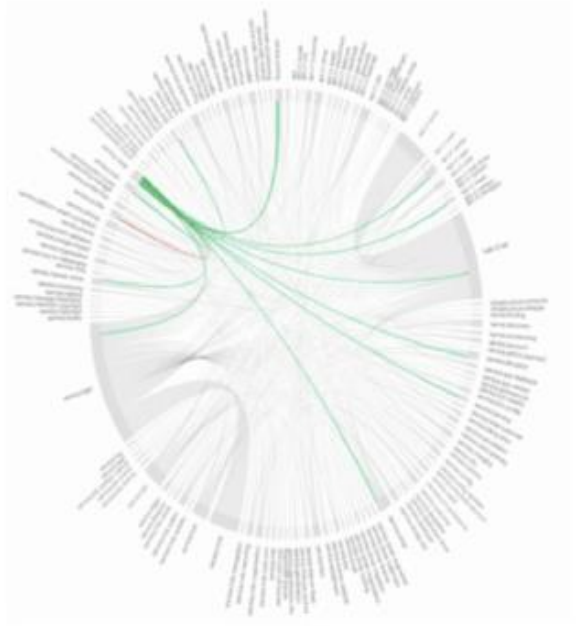
- **SOA - servisne orientovaná architektúra**
  - Ako stanoviť veľkosť služby
  - Zlyhanie služby
  - Škálovanie špecifických častí služby

- **Microservice**
  - Efektívna škálovateľnosť aplikácií
  - Flexibilné aplikácie
  - Vysoký výkon aplikácií



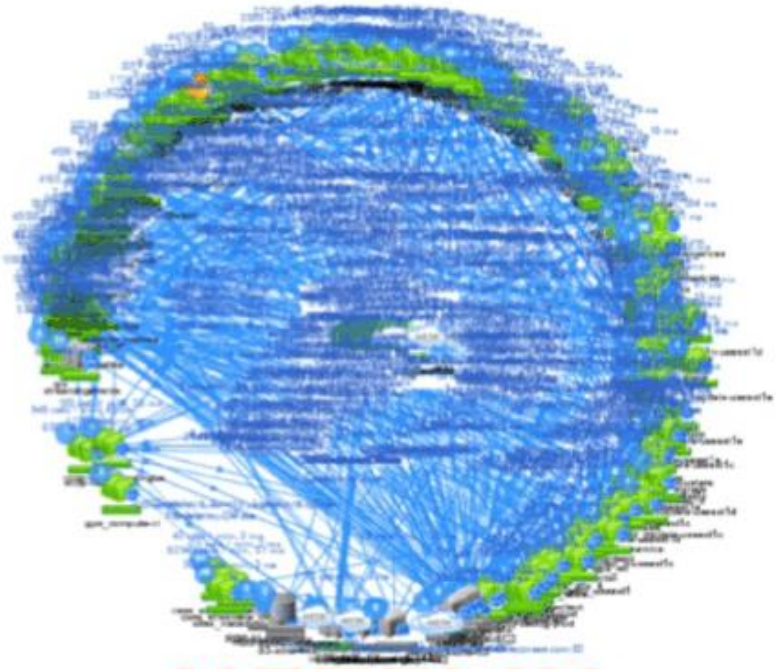
Aplikácia microservisov

450 microservices



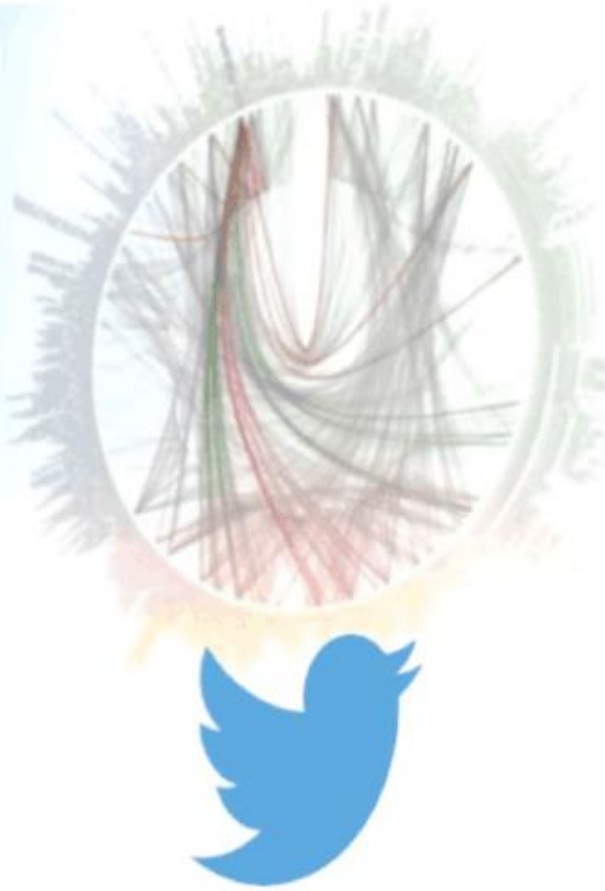
HAIL O

500+ microservices



NETFLIX

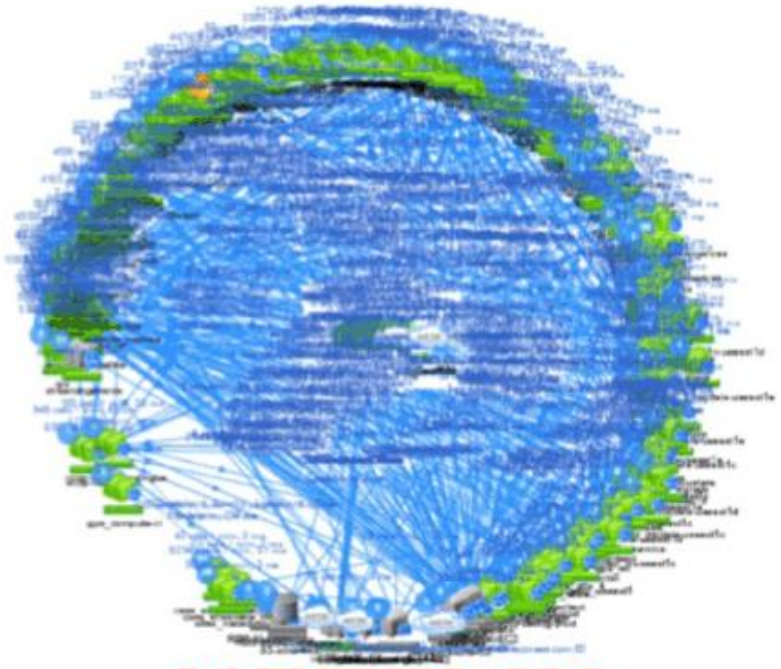
500+ microservices



Source:
Netflix: http://www.slideshare.net/BruceWong3/the-case-for-chaos
Twitter: https://twitter.com/adrianco/status/441883572618948608
Hail-o: https://sudo.hailoapp.com/services/2015/03/09/journey-into-a-microservice-world-part-3/

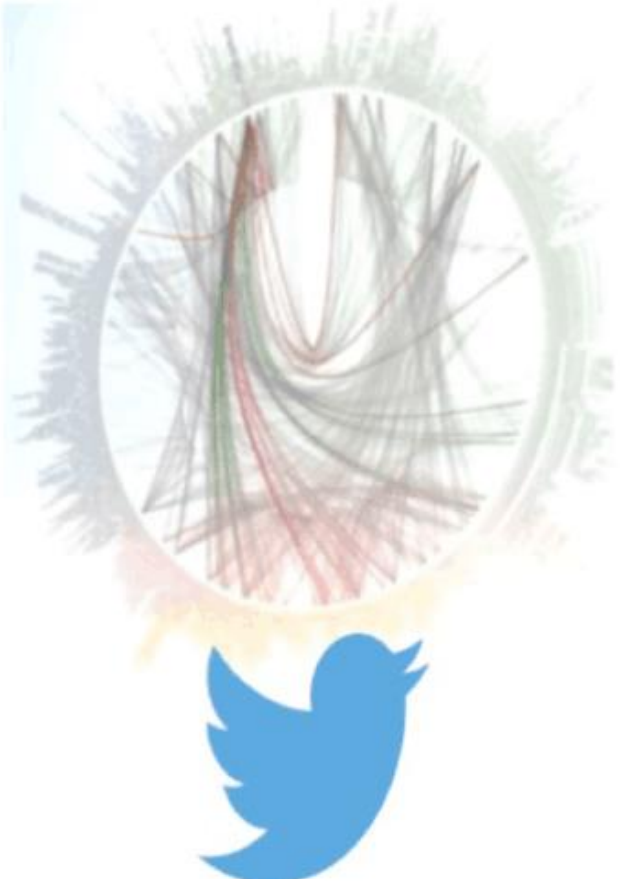
Aplikácia microservisov

500+ microservices

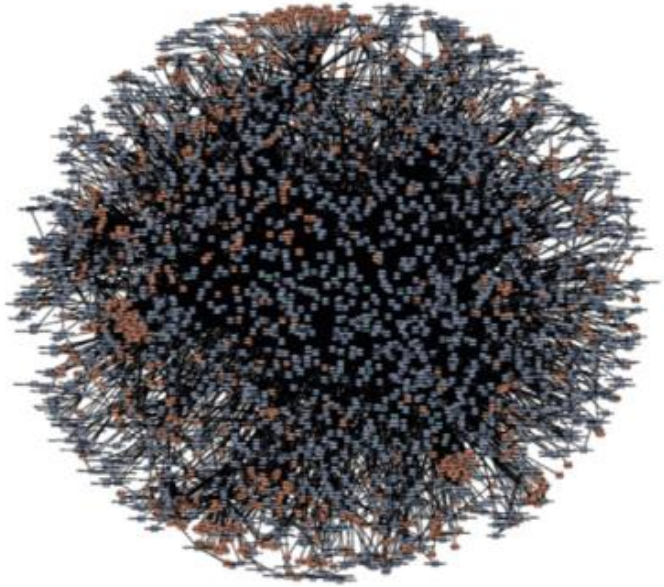


NETFLIX

500+ microservices



500+ microservices



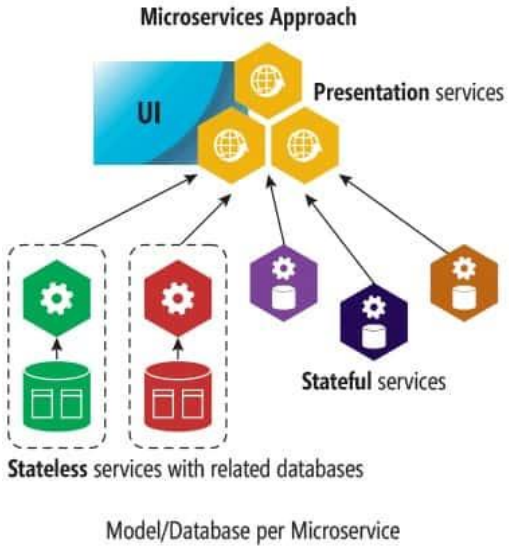
aws

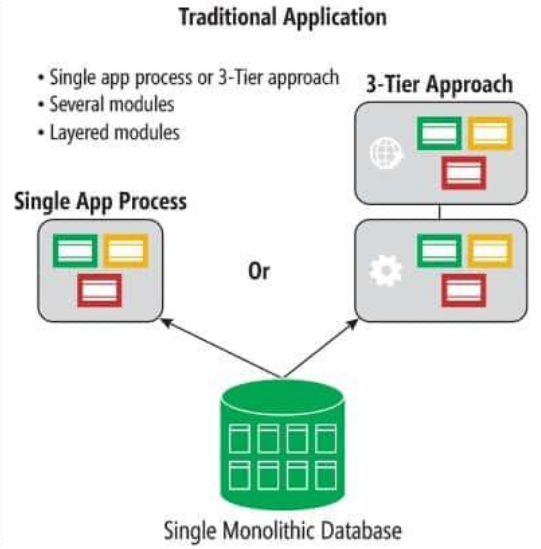
Source:
Netflix: http://www.slideshare.net/BruceWong3/the-case-for-chaos
Twitter: https://twitter.com/adrianco/status/441883572618948608
Hail-o: https://sudo.hailoapp.com/services/2015/03/09/journey-into-a-microservice-world-part-3/

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

Techniky vývoja softvéru

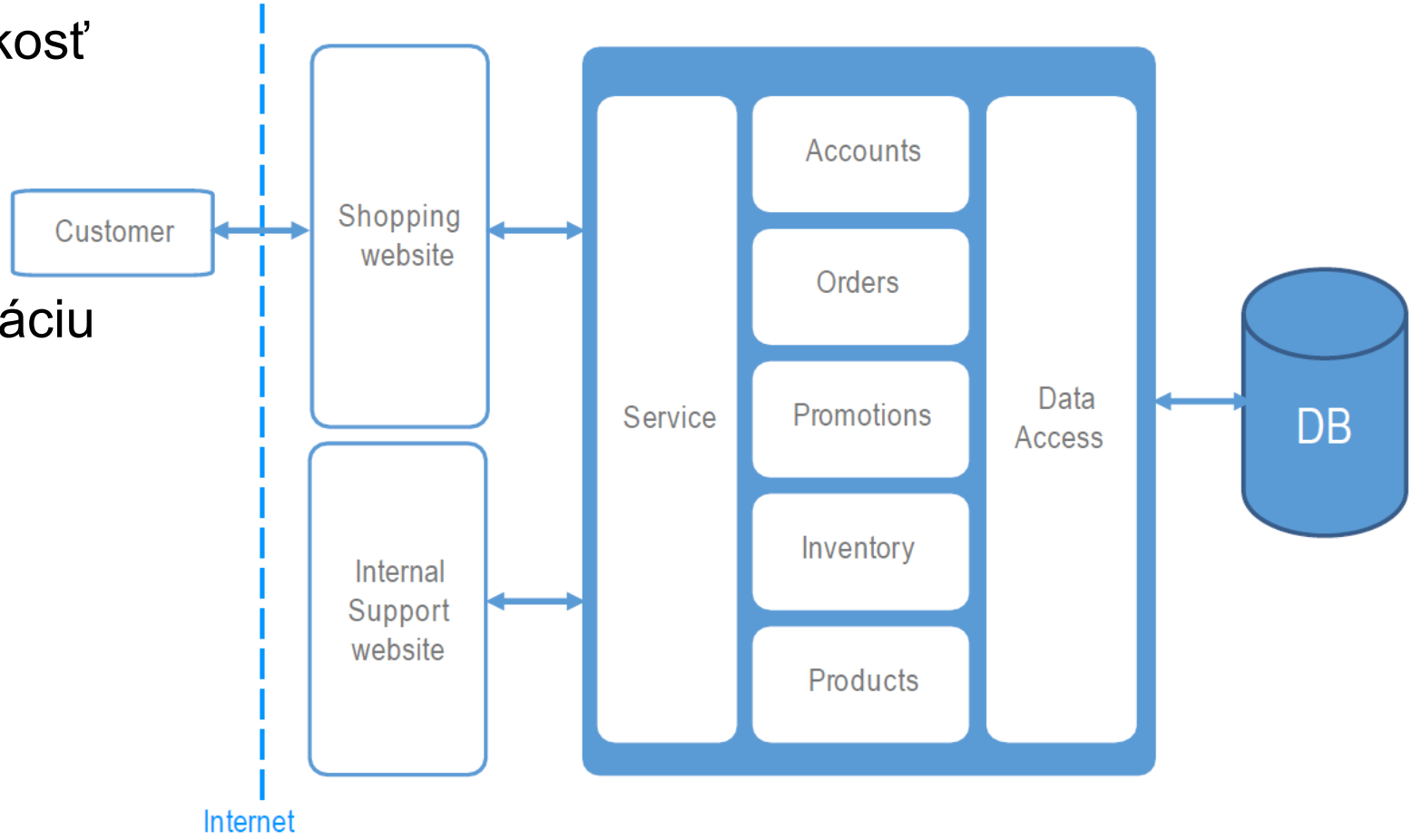
- **SOA vs Microservice**
  - Microservice je časťou SOA
  - Microservice musia byť nezávisle nasaditeľné, SOA nie
  - Klasický systém SOA je orientovaný na platformou,
  - Microservice ponúkajú viac možností vo všetkých rozmeroch
  - SOA je architektonický vzor, v ktorom komponenty aplikácií poskytujú služby iným komponentom
  - V SOA však tieto komponenty môžu patriť do tej istej aplikácie
  - V Microservice sú tieto komponenty sústavou nezávisle nasaditeľných služie





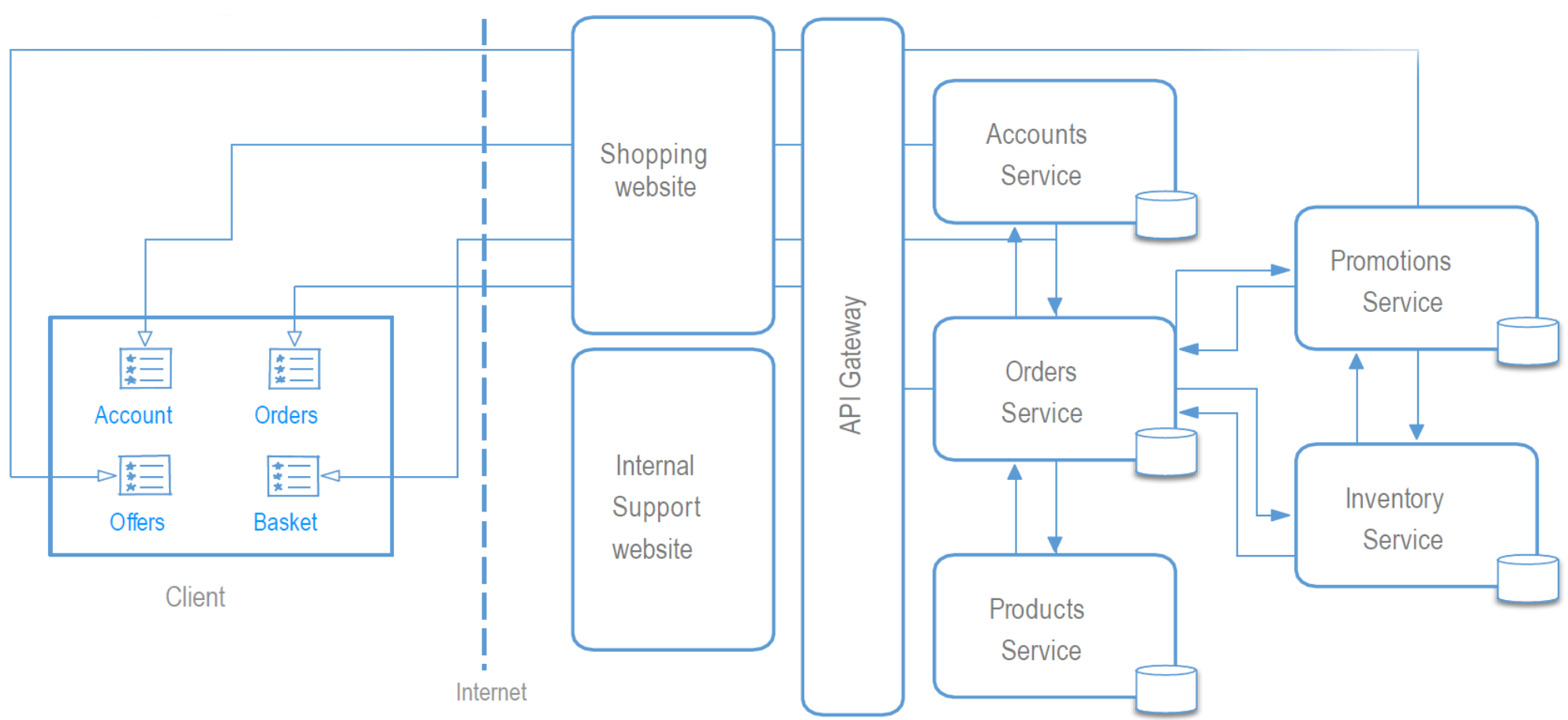
Techniky vývoja softvéru

- **SOA a monolit**
  - Žiadne obmedzenia na veľkosť
  - Dlhší vývojový čas
  - Nedostupné funkcionality
  - Vysoká miera komunikácie
  - Škálovanie vyžaduje duplikáciu
  - Malé zmeny môžu ústiť do celkovej „prerábky“



Techniky vývoja softvéru

- **Príklad Microservice**



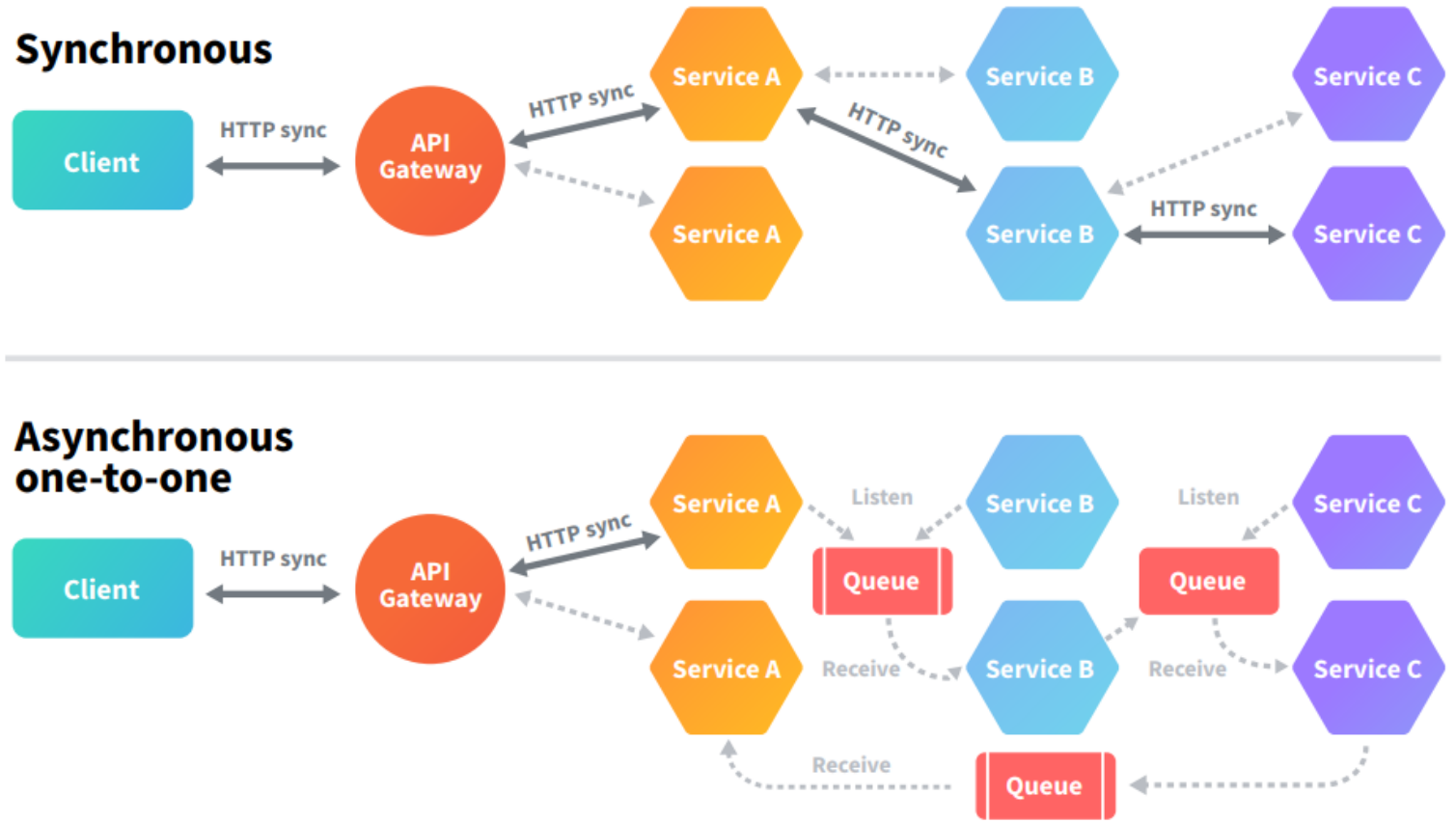
Techniky vývoja softvéru

- **Výhody Microservice:**
  - Potreba rýchlo reagovať na zmeny
  - Potreba spoľahlivosti
  - Podnikovo orientovaný návrh
  - Automatizované testovacie nástroje
  - Nástroje na vydávanie a nasadenie
  - Hostovananie na žiadosť
  - On-line cloudové služby
  - Potreba objaviť nové technológie
  - Asynchrónna komunikačná technológia
  - Jednoduchšia technológia na strane servera a na strane klienta
  - Kratšie vývojové časy
  - Spoľahlivé a rýchlejšie nasadenie
  - Umožňuje časté aktualizácie
  - Možnosť odpojenia/vymenenia častí
  - Zabezpečenie
  - Zvýšená zaťažiteľnosť
  - Rýchle riešenie problémov
  - Vysoká škálovateľnosť a lepší výkon
  - Lepšie definovateľné vlastníctvo
  - Umožňuje distribuované vývojárske tímy

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

Techniky vývoja softvéru

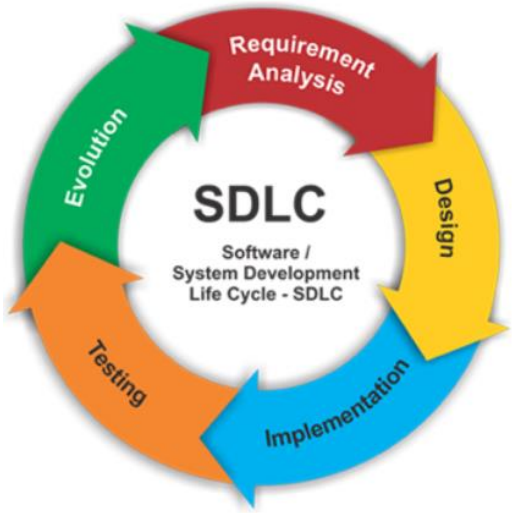
- **Komunikácia v Microservice**



Princípy životného cyklu vývoja systémov a zásady bezpečného vývoja softvéru

Životný cyklus vývoja softvéru (SDLC)

- predstavuje **súbor fáz**, ktorými softvér prechádza od návrhu až po ukončenie prevádzky:
- **Požiadavky a analýzy**
  - Zber a špecifikácia funkčných a nefunkčných požiadaviek.
  - Identifikácia bezpečnostných potrieb už v tejto fáze.
- **Návrh**
  - Vytvorenie architektúry – databázy, moduly, rozhrania.
- **Implementácia**
  - Písanie kódu podľa bezpečnostných štandardov. Dodržiavanie princípov bezpečného programovania.
- **Testovanie**
  - Overenie, či softvér odoláva útokom a nesie len akceptovateľné riziká.
- **Nasadenie**
  - Správna konfigurácia do systému a prostredia.
- **Prevádzka a údržba**
  - Monitorovanie, logovanie, reakcie na incidenty. Aktualizácie a záplaty zraniteľností.
- **Ukončenie**
  - Bezpečné vyradenie :
    - Vymazanie alebo anonymizácia údajov.
    - Odstránenie prístupov a deaktivácia služieb.



Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

Zásady bezpečného vývoja softvéru

- **Security by Design**
  - Bezpečnosť sa plánuje od začiatku, nie dodatočne
  - Navrhnutie tak, aby bol odolný voči hrozbám už na úrovni archtektúry.
- **Princíp najnižších oprávnení (Least Privilege)**
  - Používateľ má iba minimálne potrebné oprávnenia.
- **Obrana v hĺbke (Defense in Depth)**
  - Viac vrstiev bezpečnostných mechanizmov napr. firewall+autentifikácia+šifrovanie+monitorovanie
- **Validácia vstupov**
- **Bezpečnostné spracovanie výnimiek**
  - Neodhaľovať interné detaily systému (napr. chybové hlášky)
  - Chybové stavy musia byť kontrolované a zalogované
- **Šifrovanie citlivých údajov**
- **Bezpečnostné testovanie v každej fáze**

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

Metódy testovania a vyhodnocovania bezpečnosti systémov

Načo vôbec testy?

- Testovanie softvéru vzniklo s vývojom softvéru
- Zvyšovanie výkonu techniky – nárast zložitosti systémov a ich testovania
- Neexistencia statického stavu testov:
  - Vývoj nových technológií
  - Nových programovacích jazykov
  - Nových prístupov
  - Využívanie mobilnej techniky, cloudu, virtualizácia
- Je neoddeliteľnou súčasťou vývojového cyklu softvéru

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

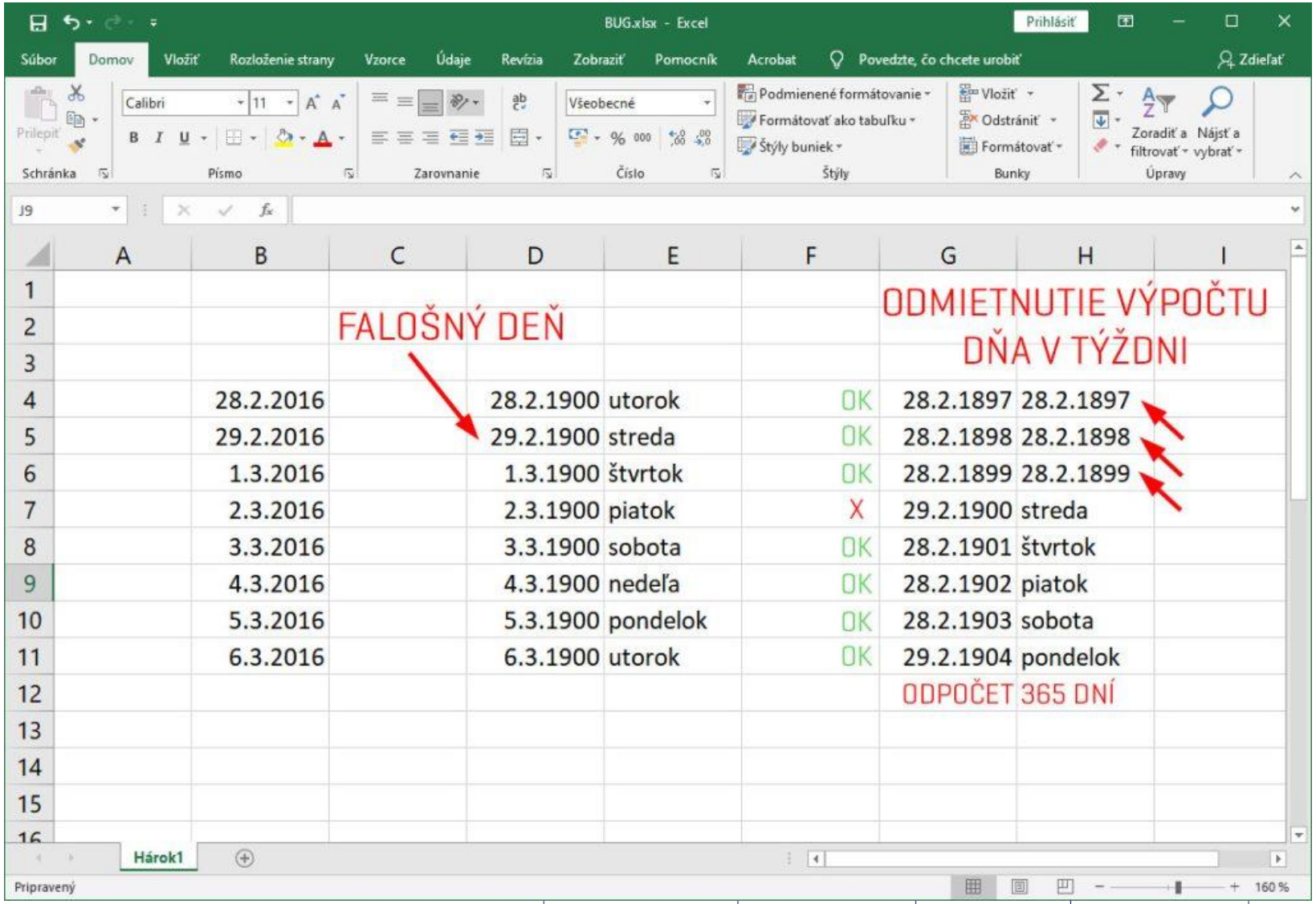
Príklad „bugu“ ktorý je aj v štandarde a bežne v praxi

- Excel
  - kompatibilita s Lotus 1-2-3
  - neexistujúci deň 29.2.1900
- *Priestupný rok totiž nastáva každé štyri roky. Presnejšie, nastáva v každom roku, ktorý je deliteľný štvorkou, s výnimkou okrúhlych dátumov storočí (ako 1900, 2000, 2100), ktoré sú priestupnými rokmi iba v prípade, že sú deliteľné 400*.

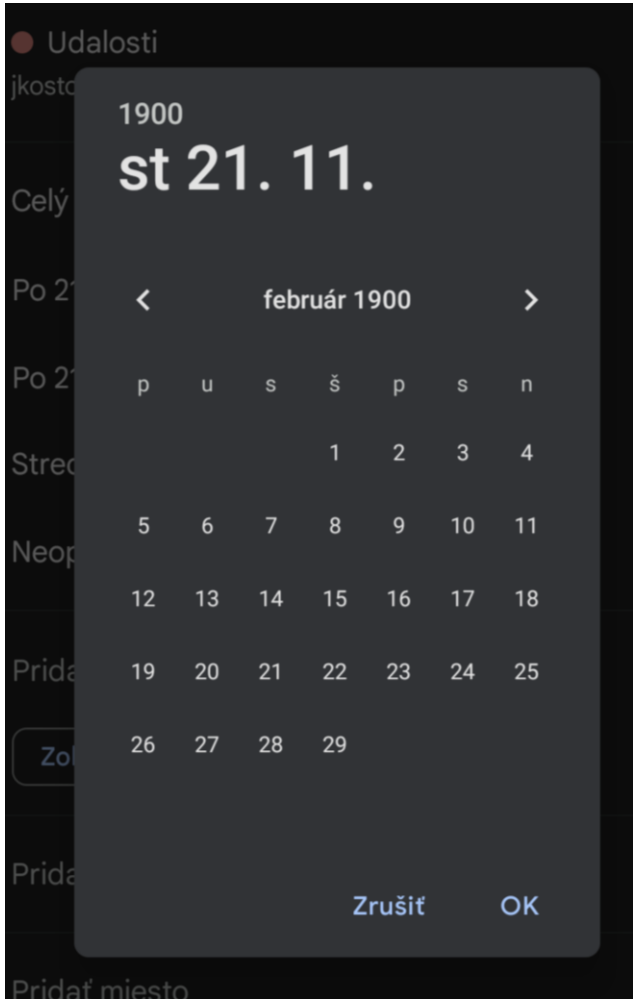
- bug je súčasťou medzinárodnej špecifikácie ISO a ECMA (European Computer Manufacturers Association),
  - bug štandardizovaný v rámci kancelárskeho Open XML formátu dokumentov

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

Príklad „bugu“ ktorý je aj v štandarde a bežne v praxi

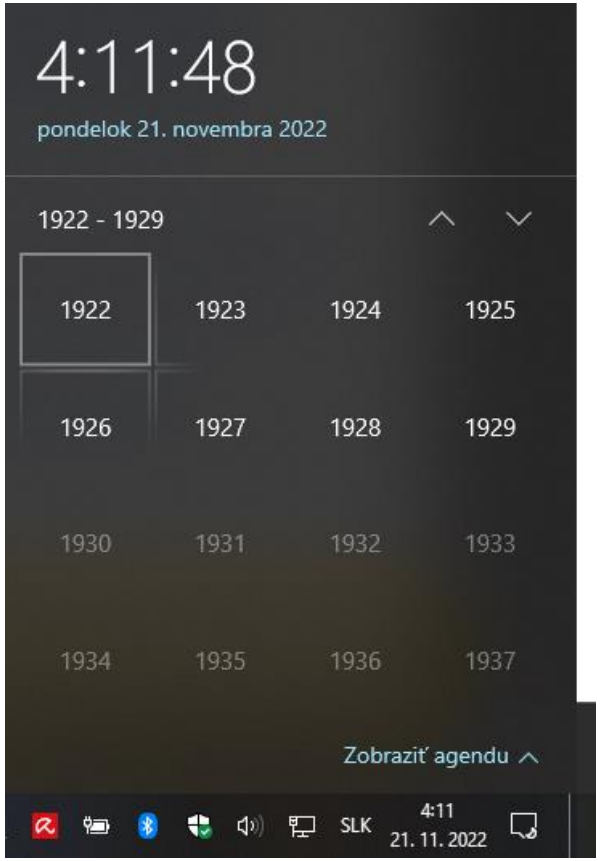


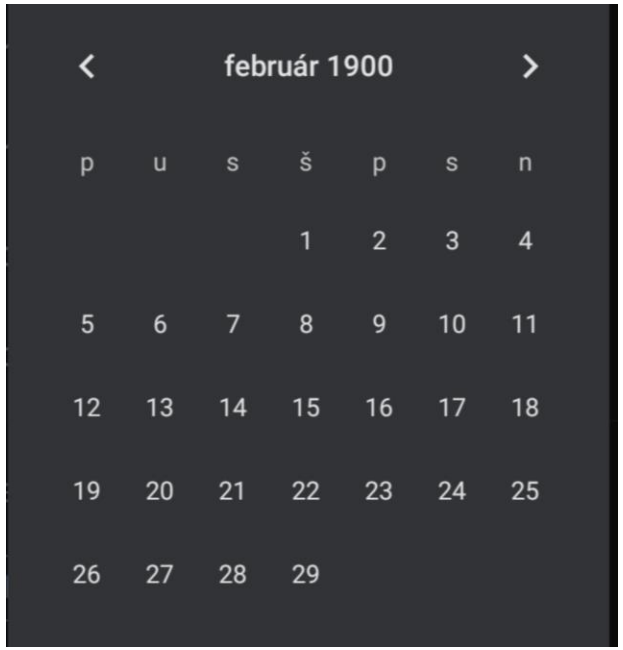
	A	B	C	D	E	F	G	H	I
1							ODMIETNUTIE VÝPOČTU		
2			FALOŠNÝ DEŇ				DŇA V TÝŽDNI		
3									
4		28.2.2016		28.2.1900	utorok	OK	28.2.1897	28.2.1897	
5		29.2.2016		29.2.1900	streda	OK	28.2.1898	28.2.1898	
6		1.3.2016		1.3.1900	štvrtok	OK	28.2.1899	28.2.1899	
7		2.3.2016		2.3.1900	piatok	X	29.2.1900	streda	
8		3.3.2016		3.3.1900	sobota	OK	28.2.1901	štvrtok	
9		4.3.2016		4.3.1900	nedeľa	OK	28.2.1902	piatok	
10		5.3.2016		5.3.1900	pondelok	OK	28.2.1903	sobota	
11		6.3.2016		6.3.1900	utorok	OK	29.2.1904	pondelok	
12							ODPOČET 365 DNÍ		
13									
14									
15									
16									

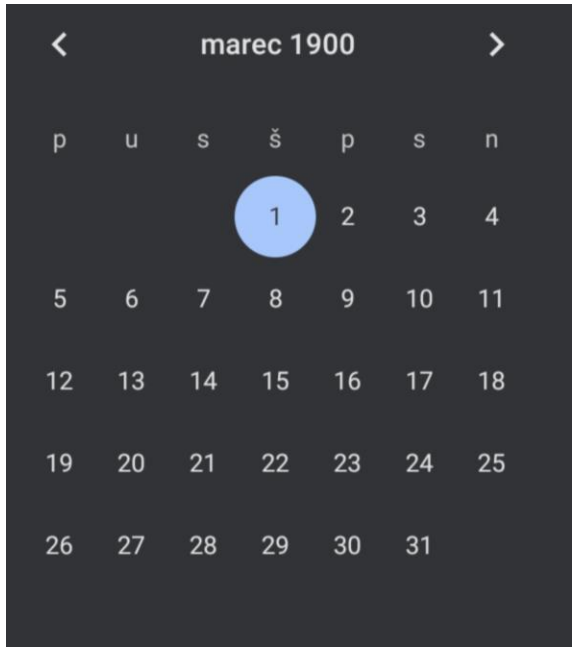


Ako to opraviť?

- ala Microsoft
  - nezobraziť daný dátum
- Android síce zobrazuje daný deň ale pri výbere to opraví







Čo potrebujeme vedieť ...

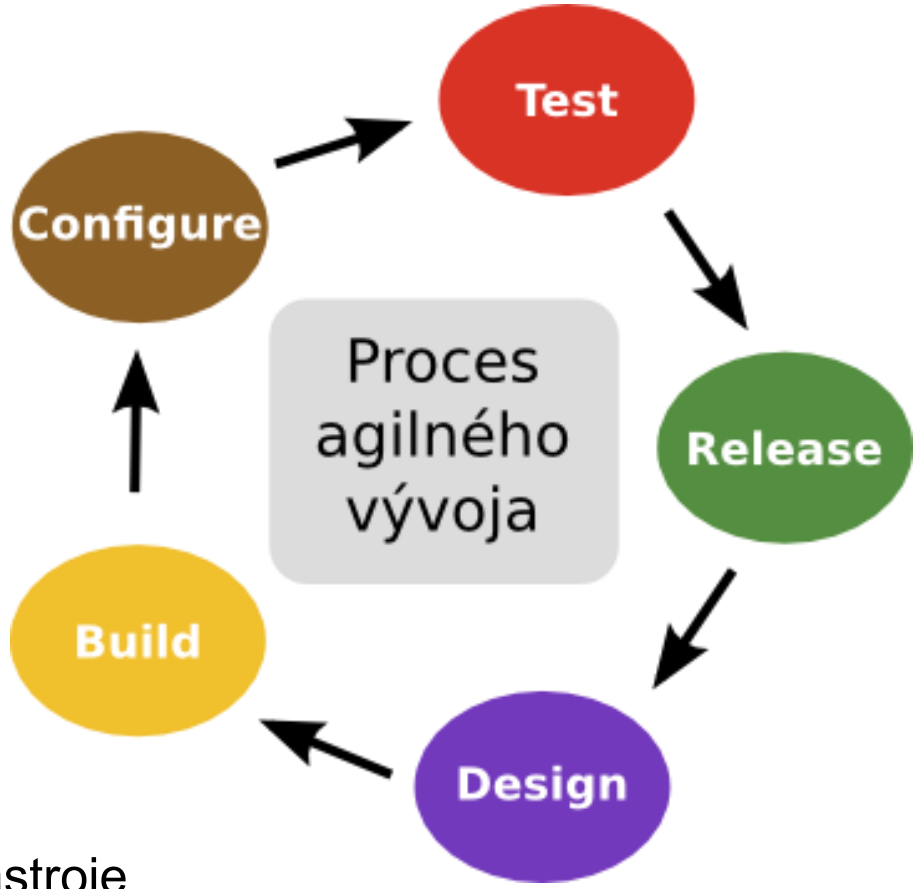
- **Požiadavky**:
  - prianie zákazníka popisujúce funkcie a vlastnosti vytváraného systému
- Základné typy:
  - **Funkčné** požiadavky – čo by to malo robiť
  - **Nefunkčné** požiadavky – vlastnosti, obmedzenia

- Nie vždy sú definované správne

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

Spôsob vývoja a testovanie

- **Tradičný spôsob**
  - Štúdia realizovateľnosti
  - Špecifikácia požiadaviek na SW
  - Návrhový dokument



- **Agilný vývoj**
  - Adaptabilita, flexibilita a prispôsobivosť na zmeny
  - Manifest:
    - Ľudia a ich komunikácia sú dôležitejší ako procesy a nástroje
    - Fungujúci softvér je dôležitejší než prepchatá dokumentácia
    - Spolupráca so zákazníkom je v popredí pred vyjednávaním zmluvy
    - Je dôležitejšie reagovať na zmeny ako dodržiavať plán
  - Frameworky: Scrum, Extreme programing, …

Základný koncept kvalita softvéru a testovanie

- Kvalitný vs. Nekvalitný ? **Bezpečný vs. Nebezpečný**?

- Rôzne pohľady:
  - Transcendentálny
  - Používateľský
  - Výrobný
  - Produkčný
  - Cenový

Základný koncept kvalita softvéru a testovanie

- Kvalitný vs. Nekvalitný ? **Bezpečný vs. Nebezpečný**?

- Rôzne pohľady:
  - Transcendentálny
  - Používateľský
  - Výrobný
  - Produkčný
  - Cenový

- Podľa normy ISO/IEC 25010:
  - **Miera v akej SW spĺňa stanovené a implicitné potreby**, pokiaľ je používaný na navrhnutý účel
  - Základné charakteristiky:
    - Funkčnosť
    - Účinnosť
    - Kompatibilita
    - Použiteľnosť
    - Bezporuchovosť
    - Bezpečnosť
    - Udržateľnosť
    - Prenositeľnosť

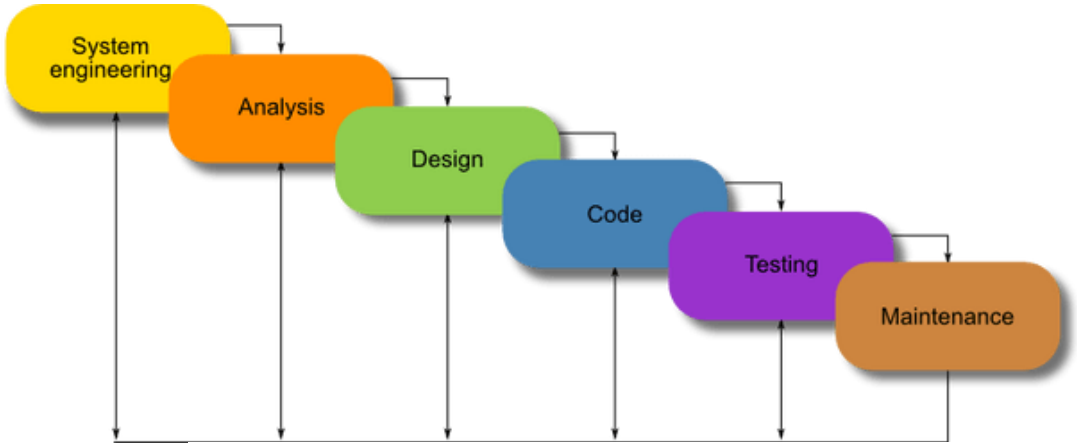
Metódy testovania a vyhodnocovania

- Čo je to testovanie?
  - Proces riadeného **spúšťania** softvérového produktu s cieľom zistiť či spĺňa špecifikované alebo implicitné potreby používateľa

- Slúžia na:
  - Identifikáciu zraniteľností.
  - Preverenie odolnosti voči útokom.
  - Hodnotenie súladu so štandardami a bezpečnostnými politikami.

- Delia sa na kategórie podľa:
  - Prístupu k systému.
  - Účelu.
  - Hĺbky testovania.

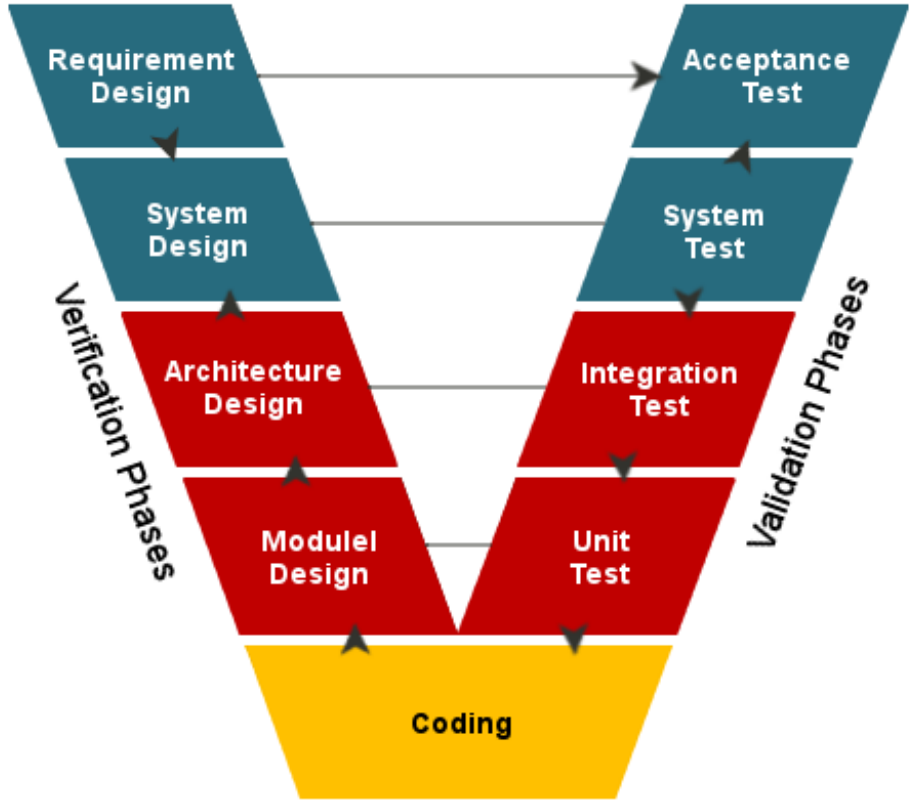
Verifikácia a validácia



- Verifikácia:
  - **vytvárame produkt správne?**
  - Požiadavky, návrh, zdrojový kód, integrácia
  - Testovanie – forma dynamickej verifikácie

- Validácia:
  - **vytváram správny produkt?**
  - Účasť zákazníka, overenie funkčnosti a poskytnutia služieb
  - Akceptačné testovanie
    - dodaný SW robí to čo má

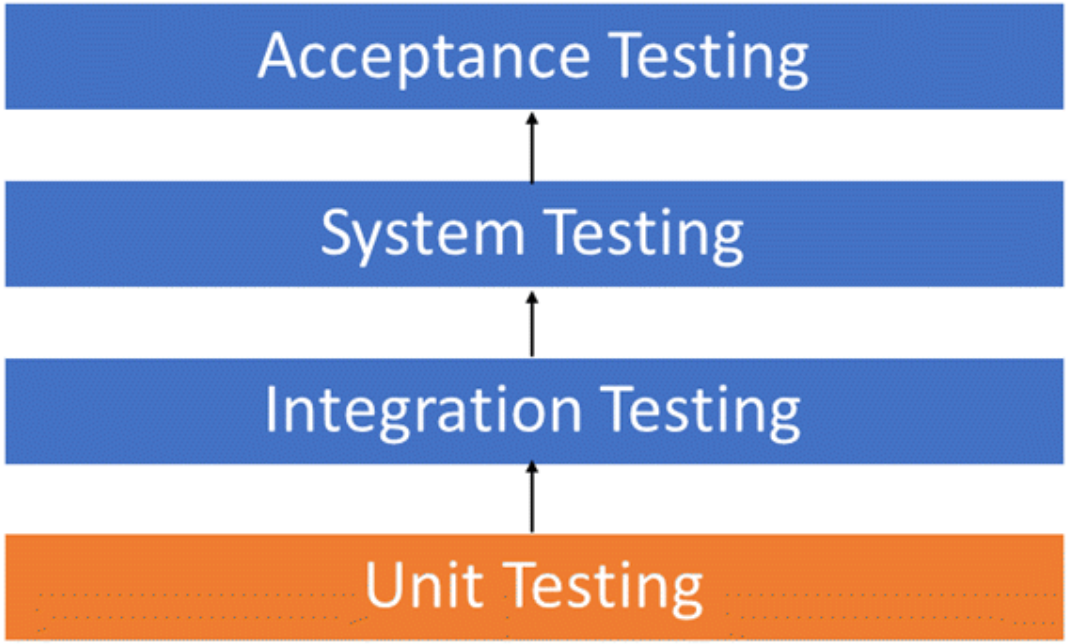
- Testovací V-model pre ilustráciu VaV



Typy testov podľa úrovne

- Rôzne typy testov na rôznych úrovniach vývoja softvéru

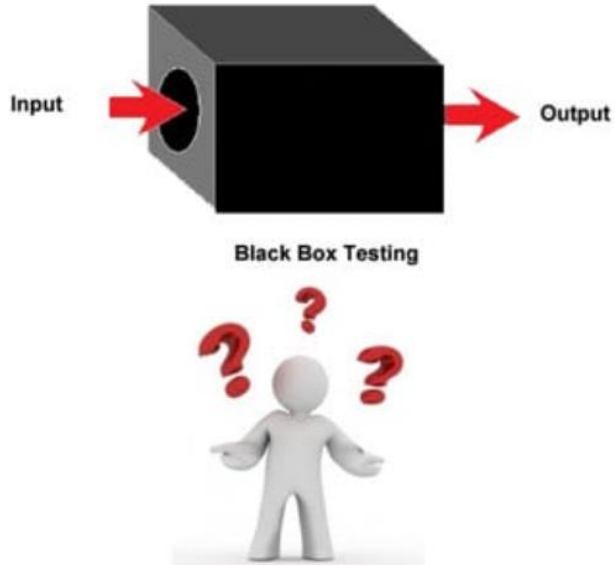
- 4 úrovne testovania:
  - Testovanie komponentov (*Unit testing*)
  - Integračné testovanie (*Integration Testing*)
  - Systémové testovanie (*System testing*)
  - Akceptačné testovanie (*Acceptance testing*)



Delenie podľa prístupu k systému

- **White-box testovanie**
  - Tester má úplný prístup ku kódu, konfiguráciám, architektúre.
  - Metódy:
    - SAST (Static Application Security Testing)
    - Bezpečnostný audit
    - Zdrojová analýza kódu
    - Fuzzing s prístupom k internému kódu

- **Black-box testovanie**
  - Tester nevie nič o vnútornej architektúre systému a simuluje skutočného útočníka.
  - Metódy:
    - Penetračné testovanie
    - DAST (Dynmic Application Security Testing)
    - Vulnerability scanning

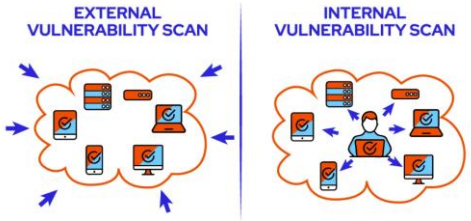


- **Gray-box testovanie**
  - Tester má čiastočné informácie (napr. prístupové údaje, architektúru), ide o kombináciu bieleho a čierneho boxu.
  - Metódy:
    - Penetračné testovanie
    - IAST (Interactive Application Security Testing)

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

Delenie podľa účelu testovania

- **Detekcia zraniteľností**
  - Cieľ: identifikovať slabiny skôr, ako ich zneužije útočník
  - Metódy:
    - Vulnerability scanning
    - SAST/DAST/SCA
    - Fuzzing
    - Bezpečnostný audit



- **Overenie odolnosti a reakcie**
  - Cieľ: zistiť, ako systém reaguje na reálne útoky
  - Metódy:
    - Penetračné testovanie
    - Simulácia útokov (napr. phishing)
    - Incident response testing (Blue Team)
- **Overenie súladu s normami (compliance testing)**
  - Cieľ: posúdiť súlad s legislatívou, normami a štandardmi.
  - Metódy:
    - Bezpečnostný audit.
    - Gap analýza (porovnanie reality a požiadaviek).
    - Kontrolné chcecklisty.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

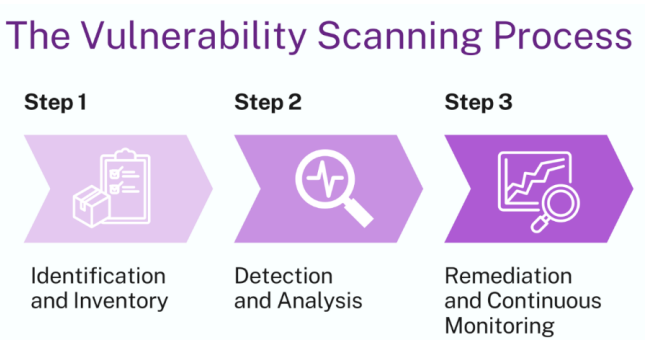
Delenie podľa účelu testovania

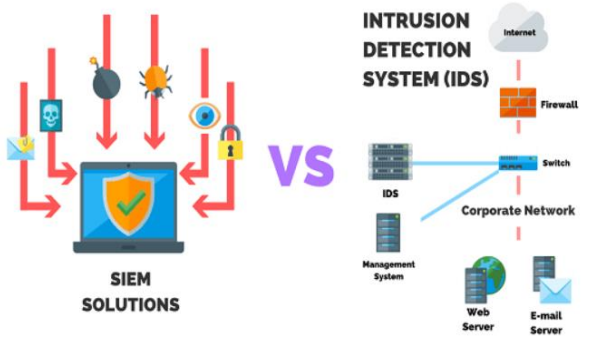
- De
  -
  -
- Ov
  -
  -
- Ov
  -
  -

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

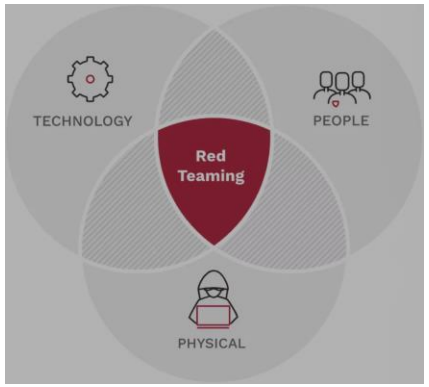
Delenie podľa hĺbky testovania

- **Povrchové testovanie (non-intrusive)** – má minimálny dopad na produkciu, bežne sa robí vo veľkých organizáciách ako prvý krok
  - Metódy:
    - Vulnerability scanning
    - Externý audit
    - Log analýza
    - SIEM/IDS monitoring





- **Hlboké testovanie (intrusive)** – môže ovplyvniť prevádzku systému, často vykonávané v testovacom prostredí.
  - Metódy:
    - Penetračné testovanie.
    - Red teaming.
    - Fuzzing.
    - Detailná analýza kódu.



Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

Metódy vyhodnocovania bezpečnosti systémov

- Slúžia na systematické posúdenie úrovne ochrany systémov, údajov a infraštruktúry pred rôznymi typmi hrozieb.Ich účelom je:
- **Identifikácia zraniteľností**
  - Odhalia slabé miesta v systéme, ktoré by mohol útočník zneužiť.
  - Príklady: nezabezpečené porty, slabé heslá, chybné konfigurácie.
- **Overenie efektivity bezpečnostných opatrení**
  - Zistia, či implementované opatrenia (napr. firewall, šifrovanie, autentifikácia) reálne fungujú a chránia systém tak, ako majú.
- **Simulácia reálnych útokov a reakcií systému**
  - Umožňujú preveriť, ako systém reaguje na pokusy o prienik či zneužitie.
  - Pomáhajú zlepšiť detekčné a reakčné schopnosti (napr. pomocou SIEM, IDS/IPS)
- **Hodnotenie súladu s normami a legislatívou**
  - Overujú, či systém spĺňa požiadavky bezpečnostných štandardov a predpisov (ISO/IEC27001, NIST, GDPR, HIPAA...)
- **Riadenie rizík (risk managemet)**
  - Pomáhajú určiť pravdepodobnosť hrozby a dopad na systém a organizáciu.
  - Umožňuje prioritizovať, čo je potrebné opraviť ako prvé.
- **Pripravenosť organizácie**
  - Testovanie ukáže slabé miesta nielen v systéme, ale aj postupoch, ľudských faktoroch a procesoch.
- **Podpora pri plánovaní a investíciách do bezpečnosti**
  - Na základe výsledkov vie organizácia rozhodnúť, kam smerovať rozpočet a úsilie (do ochrany sietí, školenia zamestnancov, zálohovania, monitoringu...)

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

Vyhodnocovanie bezpečnosti – kvalitatívne metódy

- na základe skúseností, odborného posúdenia.
  - **Matica rizík** – riziká sa hodnotia na základe pravdepodobnosti a dopadu pomocou slovného opisu. (Napr. pravdepodobnosť: nízka/stredná/vysoká, dopad: malý/stredný/kritický).
    - Výsledkom je zaradenie rizika do zón (napr. zelená – prijateľná, červená – kritická).
  - **Brainstorming** – hodnotenie je založené na skúsenostiach, nie dátach.
  - **Checklisty a kontrolné zoznamy** – používajú sa na overenie dodržiavania bezpečnostných politík alebo štandardov.
  - **SWOT analýza bezpečnosti** – posúdenie silných a slabých stránok, príležitostí a ohrozené systému z pohľadu bezpečnosti.
  - **Kvalitatívne hodnotenie dopadu incidentov**
    - Namiesto presných čísel sa opisujú dôsledky ako: *strata dôveryhodnosti, poškodenie reputácie, právne dôsledky.*

Vyhodnocovanie bezpečnosti – kvantitatívne metódy

- Používajú číselné údaje, ako pravdepodobnosť, štatistiky a dopad hrozieb. Kvantitatívne metódy sú presnejšie, ale náročnejšie na dáta. Patria sem:
- **Výpočet rizika** (Quantitative Risk Assessment – QRA)
  - Používa vzorec: Riziko=Pravdepodobnosť x Dopad
- **Očakávaná ročná strata** (ALE – Annual Loss Expectancy)
  - Vzorec: Očakávaná strata=Strata pri jednom incidente x očakávaný ročný výskyt
- **Cost-Benefit Analysis (CBA)**
  - posudzuje sa, či sa oplatí bezpečnostné opatrenie na základe porovnania nákladov na implementáciu vs. očakávaných strát bez opatrenia.
- **Štatistická analýza historických incidentov**
  - Používa sa na predikciu budúcich hrozieb. Vychádza z analýzy údajov o minulých bezpečnostných udalostiach.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

Metodiky a štandardy k zabezpečeniu bezpečnosti

- Slúžia ako osvedčené a uznávané rámce, ktoré organizáciám pomáhajú systematicky riadiť, hodnotiť a zlepšovať bezpečnosť svojich systémov, dát a procesov.
- **Slúžia na:**
  - Zavedenie osvedčených postupov
  - Zjednotenie prístupu k bezpečnosti
  - Posúdenie a zlepšovanie bezpečnostnej úrovne
  - Zníženie rizika a ochrana pred hrozbami
  - Podpora certifikácie a dôveryhodnosti.
- **Medzi najznámejšie patria:**
  - ISO/IEC 27001, 27002 (manažment bezpečnosti)
  - NIST SP 800-53, 800-115 (kontroly a testovanie bezpečnosti)
  - OWAPS Testing Guide (pre webové aplikácie)
  - Common Criteria (ISO/IEC 15408) – pre certifikáciu bezpečnostných produktov.

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

Automatizovanie testovania

- výhody zavedenia automatizácie do testovania:

  - výrazné zníženie nákladov na údržbu vďaka automatizácii regresných testov
  - vysoké pokrytie regresnými testami kvôli zvýšenému počtu automatizovaných prípadov
  - zvýšenie výkonnosti testerov a analytikov vďaka podieľaní sa na automatizácií
  - dosiahnutie vyššej účinnosti niektorých testov
  - pokiaľ sa vytvorí framework pre automatizované testy, môže byť využitý aj v budúcnosti

Automatizovanie testovania

- základný dôvod „zvýšenie efektivity“

- pri testovaní bez použitia plnej automatizácie:
  - automatizovanie generovania testovacích dát,
  - export a transformácia vstupných dát,
  - vytváranie skriptov a jednoduchých nástrojov,
  - porovnávanie výsledkov napr. v Excel tabuľke

Automatizovanie testovania

- základný dôvod „zvýšenie efektivity“

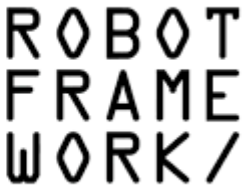
- pri testovaní bez použitia plnej automatizácie:
  - automatizovanie generovania testovacích dát,
  - export a transformácia vstupných dát,
  - vytváranie skriptov a jednoduchých nástrojov,
  - porovnávanie výsledkov napr. v Excel tabuľke

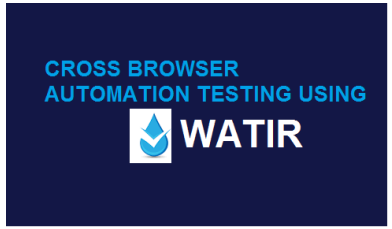
Nástroje pre automatizáciu

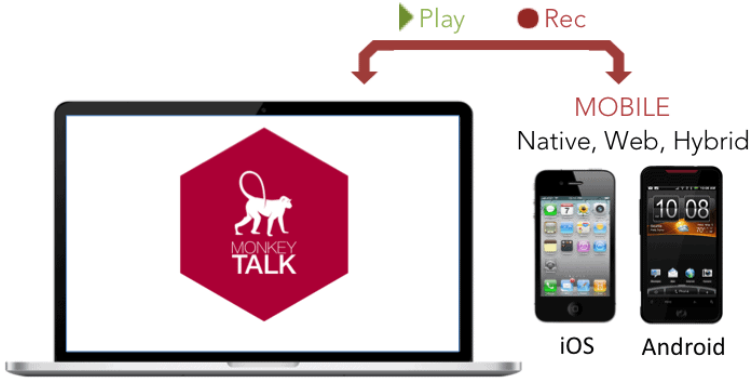
- Primárne funkčné a regresné testy:

- Platené:
  - Micro Focus Unified Functional Testing
  - Rational Functional Tester (IBM)
  - Visual Studio Test Professional
  - TestComplete – SmartBear Software

- Voľne dostupné:
  - Selenium
  - Watir
  - RobotFramework
  - MonkeyTalk – mobilné aplikácie







Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

Manažment testovania

- štruktúra testovania a procesy sú väčšinou definované vo forme dokumentácie a distribuované vo firme osobám, ktorý prichádzajú do kontaktu s testovaním

- definované býva najmä:
  - Zásady testovania
    - filozofia, hlavne ciele a celkový prístup k testovaniu v spoločnosti
  - Stratégia testovania
    - požiadavky a spôsoby akými uskutočňovať testovanie
  - Hlavný plán testovania
    - zameriava sa konkrétne na projekt a popisuje v ktorej situácii bude aplikovaná aká stratégia
  - Plán pre jednotlivé úrovne testovania
    - pri rozsiahlejších projektoch popisuje plány pre každú úroveň s detailmi pre danú úroveň (unit testy, integračné testy)

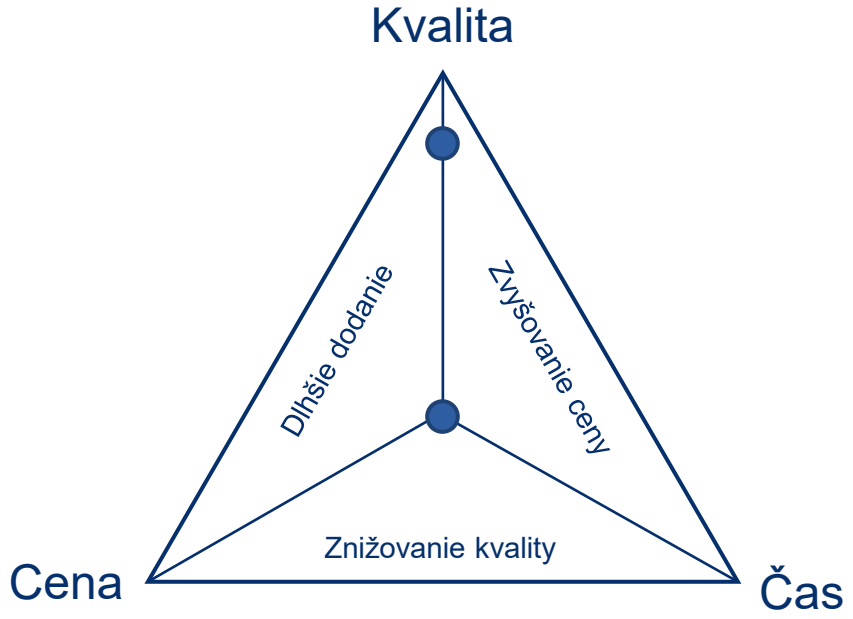
- rozsah, štruktúra, obsah a forma sa mení podľa potrieb prostredia
  - stručnejšie poznámky
  - potrebné informácie

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

Plánovanie testovania

- vychádzame z požiadaviek a potrieb „*stakeholdera*“
- základné atribúty:
  - doba dodania
  - cena
  - kvalita výsledného produktu
- ideál je nedosiahnuteľný - hľadanie kompromisu

- Trojuholník kvality
  - definuje zachovanie rovnováhy medzi atribútmi
  - pokiaľ zmeníme hodnoty jedného, ovplyvní to ostatné dva
  - napr. zníženie nákladov sa musí prejaviť v zníženej kvalite a lebo dobe dodania



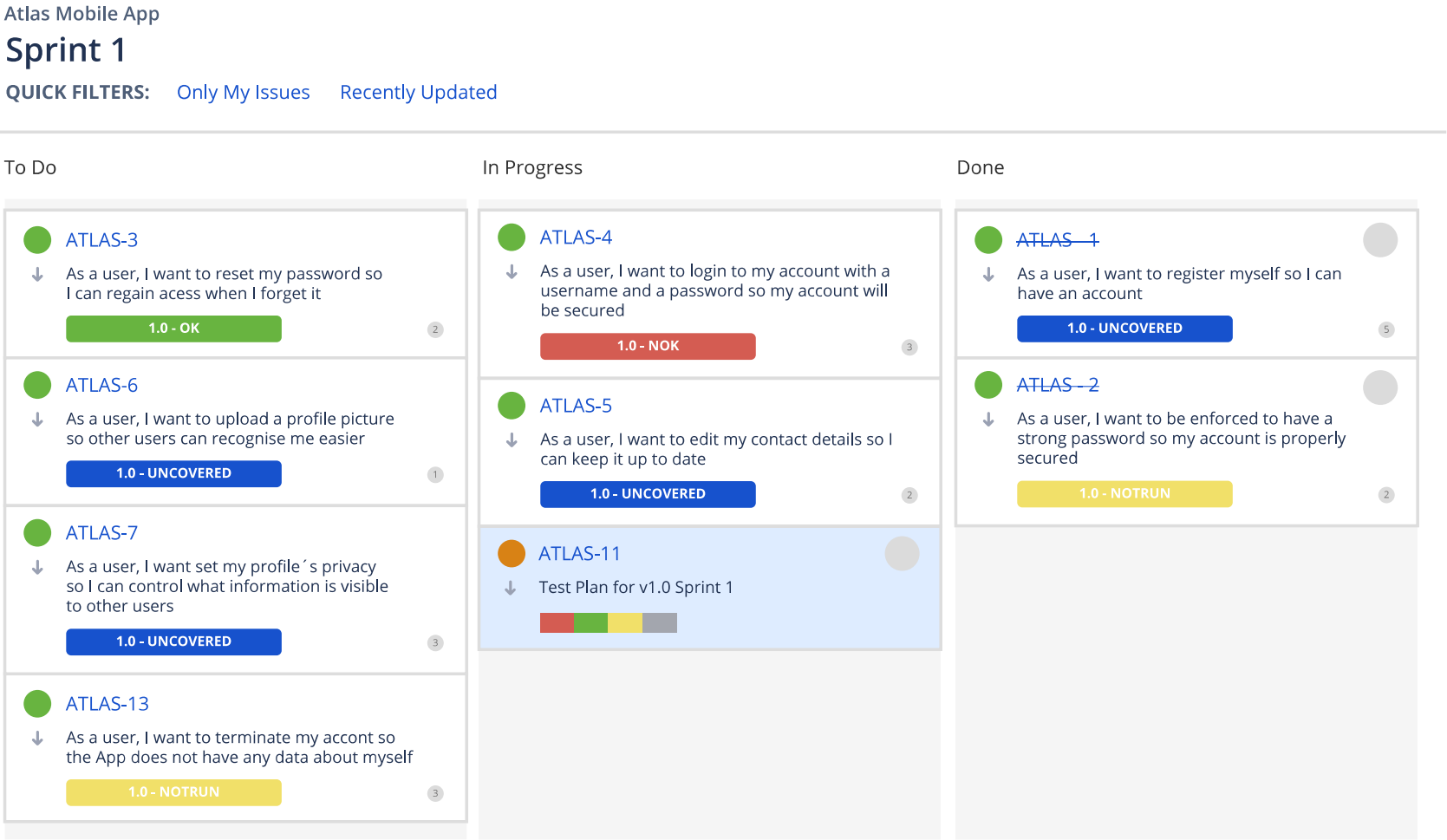
Testovací plán

- zmyslom dokumentu je definovať:
  - **Cieľ testovania** – zámer a cieľ (*dokázanie funkčnosti, splnenie požiadaviek, odladenie defektov*)
  - **Rámec testovania** – jasné vymedzenie, čo je predmetom testovania
  - **Spôsob testovania a stratégia** – ako bude testovanie prebiehať (*metódy, nástroje, prístup a technika ....*)
  - **Riziká testovacieho systému** – zmeny v systéme, zlá špecifikácia, api
  - **Potrebné zdroje** – hardvér, ľudia, požiadavky na prostredie
  - **Testovacie artefakty** – testovacie prípady, testovacie údaje, skripty, zaznamenané defekty, hlásenia o testoch
  - **Plánované aktivity** – rozvrhnutie testovania, úlohy testerov
  - **Harmonogram testovania** – od analýzy, návrhu testovacích prípadov po vyhodnotenie výsledkov testov
  - **Metriky**
  - **Identifikácia výstupných kritérií** – definícia podmienok za ktorých je ich splnenie predpokladom pre dokončenie určitej aktivity testu, alebo plánu
  - **Identifikácia kritérií pre pozastavenie a obnovenie testovania** – kedy sa prerušuje testovanie (veľa defektov odhalených) alebo znova obnovuje.
  - **Riziká ohrozujúce testovanie** – rozpočet, zdroje, zmeny v časovom harmonograme

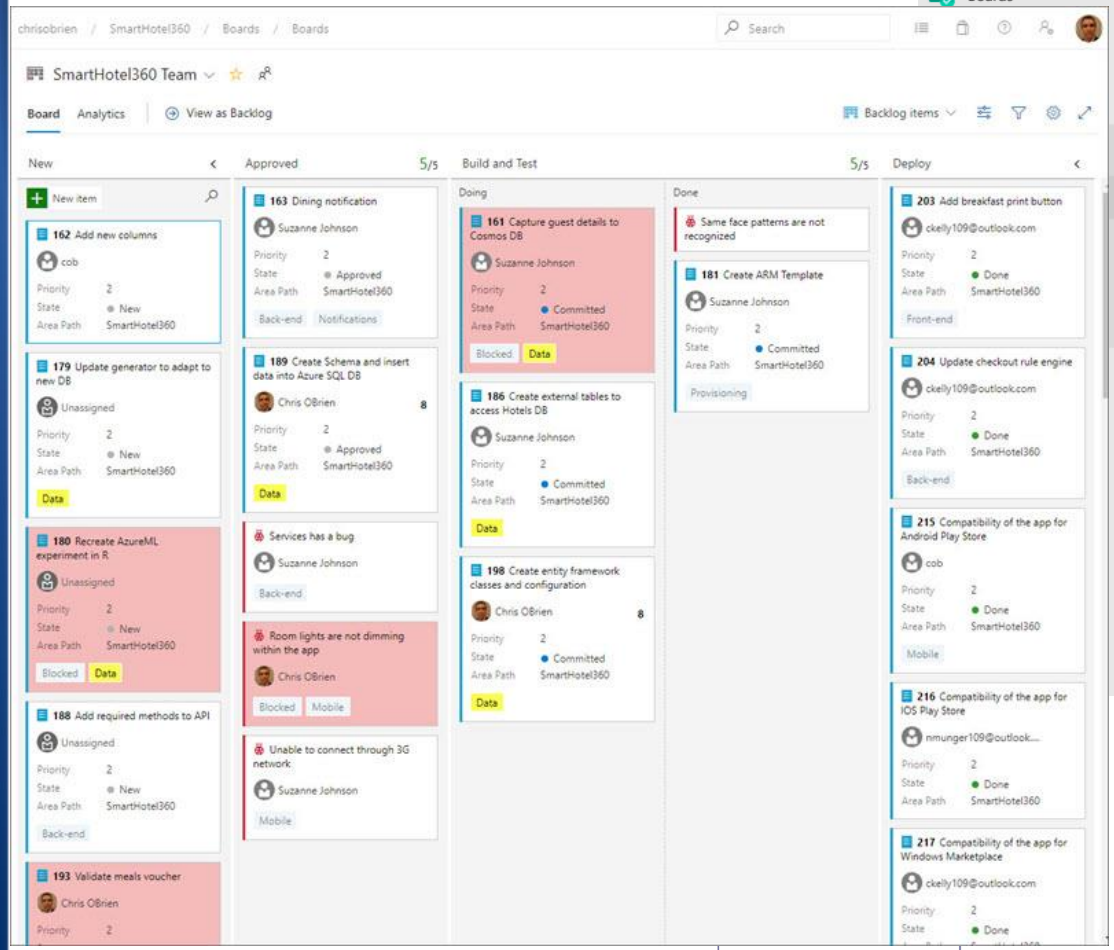
Systémy pre správu hlásenia defektov

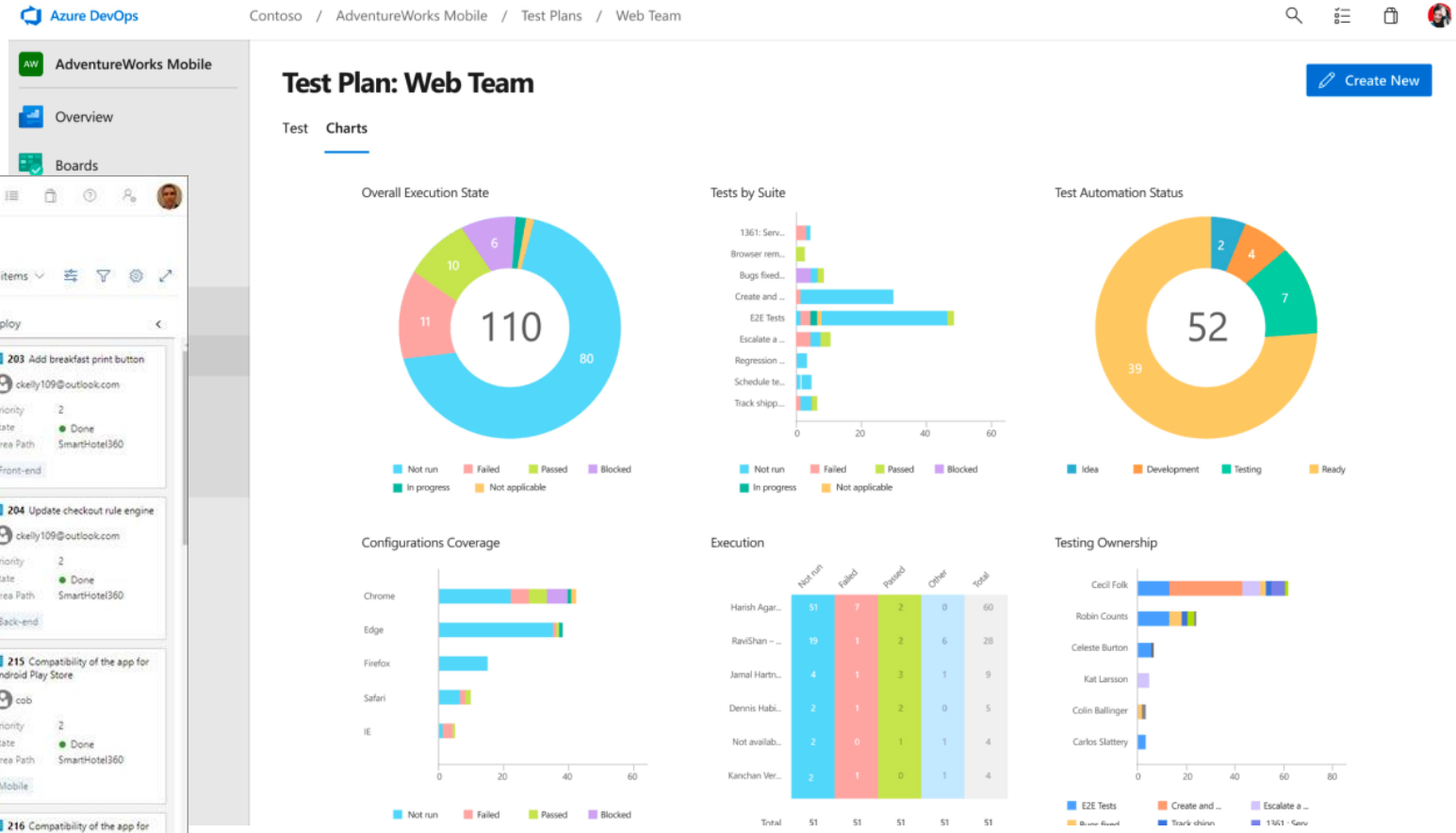
- spracovanie v textovom alebo tabuľkovom procesore
- v praxi sa používajú špecializované nástroje
- takýto systém označujeme ako *defect/issue/bug tracking system*
  - *Azure DevOps (Team Foundation Server), Backlog, Bugzilla, Mantis, Jira, Redmine....*
- bývava súčasťou systémov na riadenie testovania
  - HP Quality Center, Practitest, Testpad, TestRail, TestBench, qTest
- Systém by mal poskytovať:
  - zobrazenie stručného zoznamu hlásení, filtrovanie, zoraďovanie
  - pripojovanie príloh
  - možnosť udržiavania väčšieho množstva komentárov
  - odosielanie notifikácií pri zmene stavu
  - možnosť generovania reportov podľa metrík
  - prístup pomocou webového prehliadača

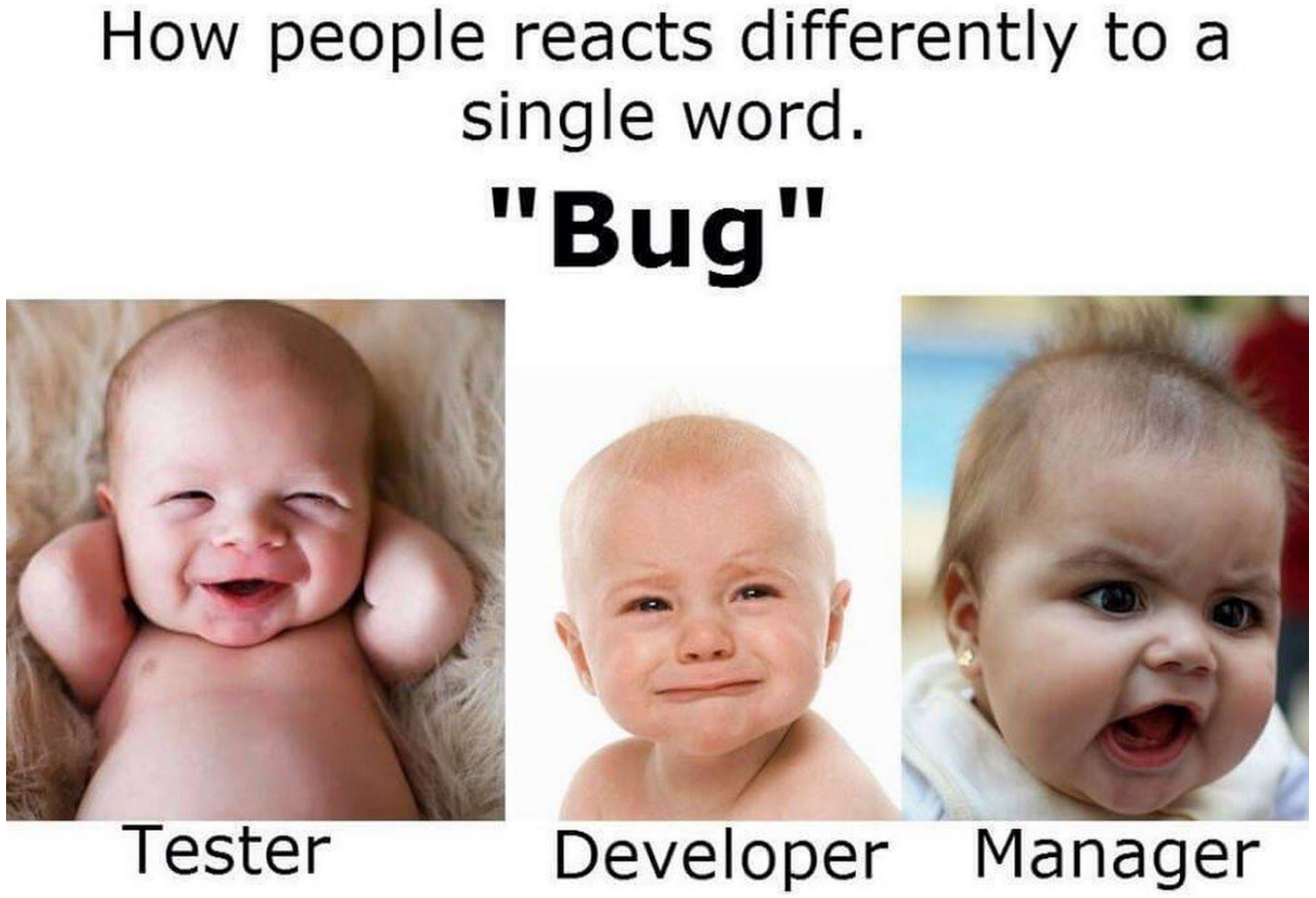
Ukážka z nástroja Xray



Ukážka z nástroja Azure DevOps







Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

Otvorená reflexia

Ktorý model vývoja softvéru kladie dôraz na testovanie v každej fáze vývoja?

A) Špirálový model

B) V-model

C) Agilný model

D) Vodopádový model

Financované Európskou úniou NextGenerationEU
PLÁN OBNOVY
MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY
KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI ŽILINSKEJ UNIVERZITY V ŽILINE

Otvorená reflexia

Čo je hlavným princípom „Security by Design“?

A) Pridávanie bezpečnosti po dokončení vývoja

B) Zameranie sa len na testovanie

C) Plánovanie bezpečnosti už od začiatku návrhu systému

D) Používanie len šifrovania

Otvorená reflexia

Aký je hlavný rozdiel medzi White-box a Black-box testovaním?

A) White-box testovanie vyžaduje prístup k internému kódu, Black-box nie

B) Black-box testovanie má prístup ku kódu

C) White-box testovanie sa zameriava len na používateľské rozhranie

D) Black-box testovanie je len manuálne

Otvorená reflexia

Ktorý z nasledujúcich princípov nepatrí medzi zásady bezpečného vývoja softvéru?

A) Princíp najnižších oprávnení

B) Obrana v hĺbke

C) Zverejňovanie interných chýb pre transparentnosť

D) Validácia vstupov

Otvorená reflexia

Aký je hlavný cieľ penetračného testovania?

A) Zlepšiť používateľské rozhranie aplikácie

B) Overiť reakciu systému na reálne útoky

C) Otestovať kompatibilitu so staršími systémami

D) Skontrolovať syntax kódu

Financované Európskou úniou
NextGenerationEU

PLÁN OBNOVY

MINISTERSTVO INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA A INFORMATIZÁCIE SLOVENSKEJ REPUBLIKY

KOMPETENČNÉ CENTRUM KYBERNETICKEJ BEZPEČNOSTI
ŽILINSKEJ UNIVERZITY V ŽILINE

Ďakujem za pozornosť

Bezpečný vývoj a testovanie softvéru

doc. Ing. Jozef Kostolný, PhD.

KC KYB UNIZA, https://kc.uniza.sk

jozef.kostolny@fri.uniza.sk