

DĚLAT  
DOBRÝ SOFTWARE  
NÁS BAVÍ

# PROFINIT

## Testování

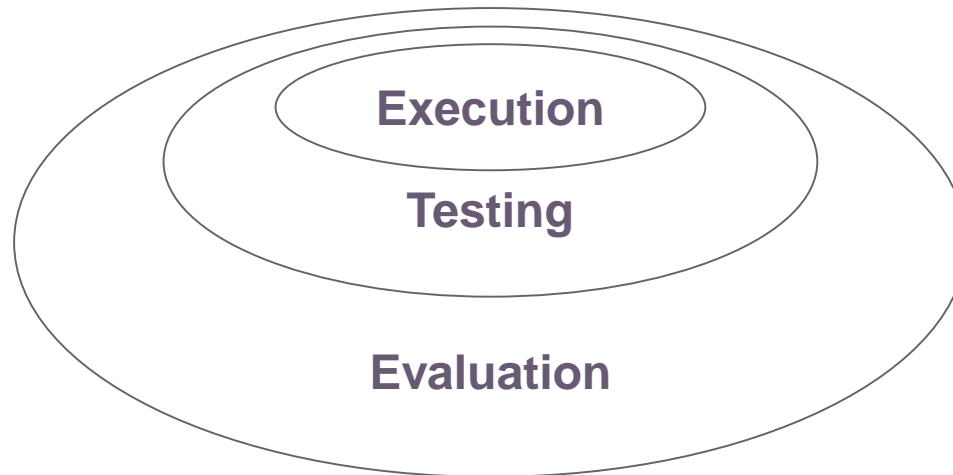
Kolektiv autorů

listopad 2018

# Schematický pohled

## Co je Software testing?

- Zkoušení / simulace provozu SW
- Ustanovení důvěry v to, že SW dělá co má, a nedělá, co nemá
- Analýza SW s cílem nalézt chyby a problémy
- Měření funkcionality a kvality SW
- Zhodnocení atributů a schopností SW, zda dosahují požadovaných či akceptovatelných výsledků
- Inspekce, stejně jako provádění testů kódu

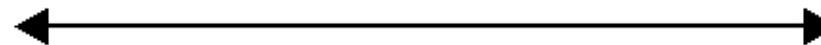


# Softwarový proces



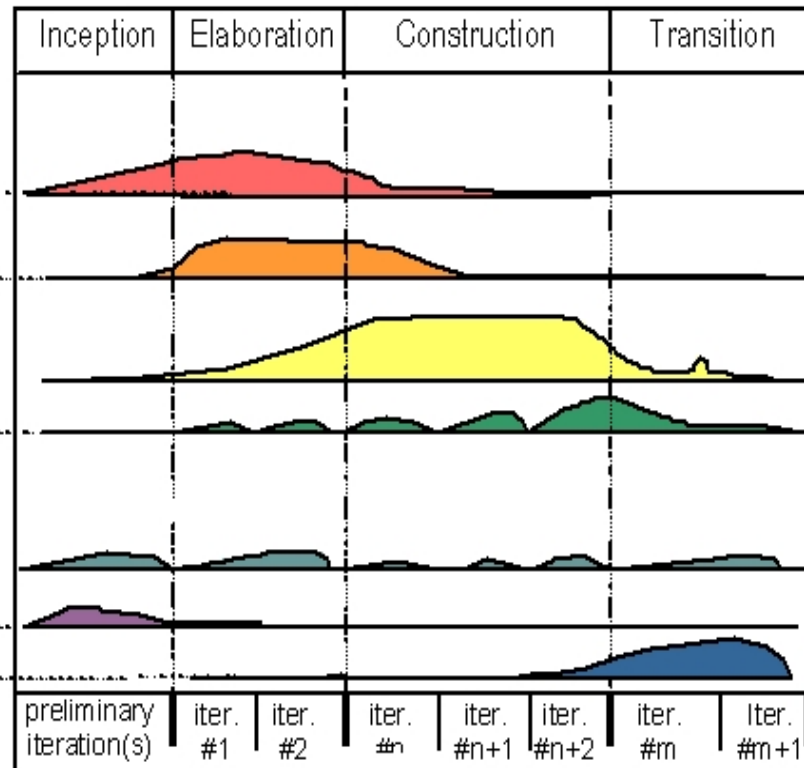
PROJECT MANAGEMENT / QUALITY ASSURANCE / DOCUMENTATION / CONFIGURATION MANAGEMENT / RELEASE MANAGEMENT / DEVOPS

Time



LCO LCA IOC

Stages

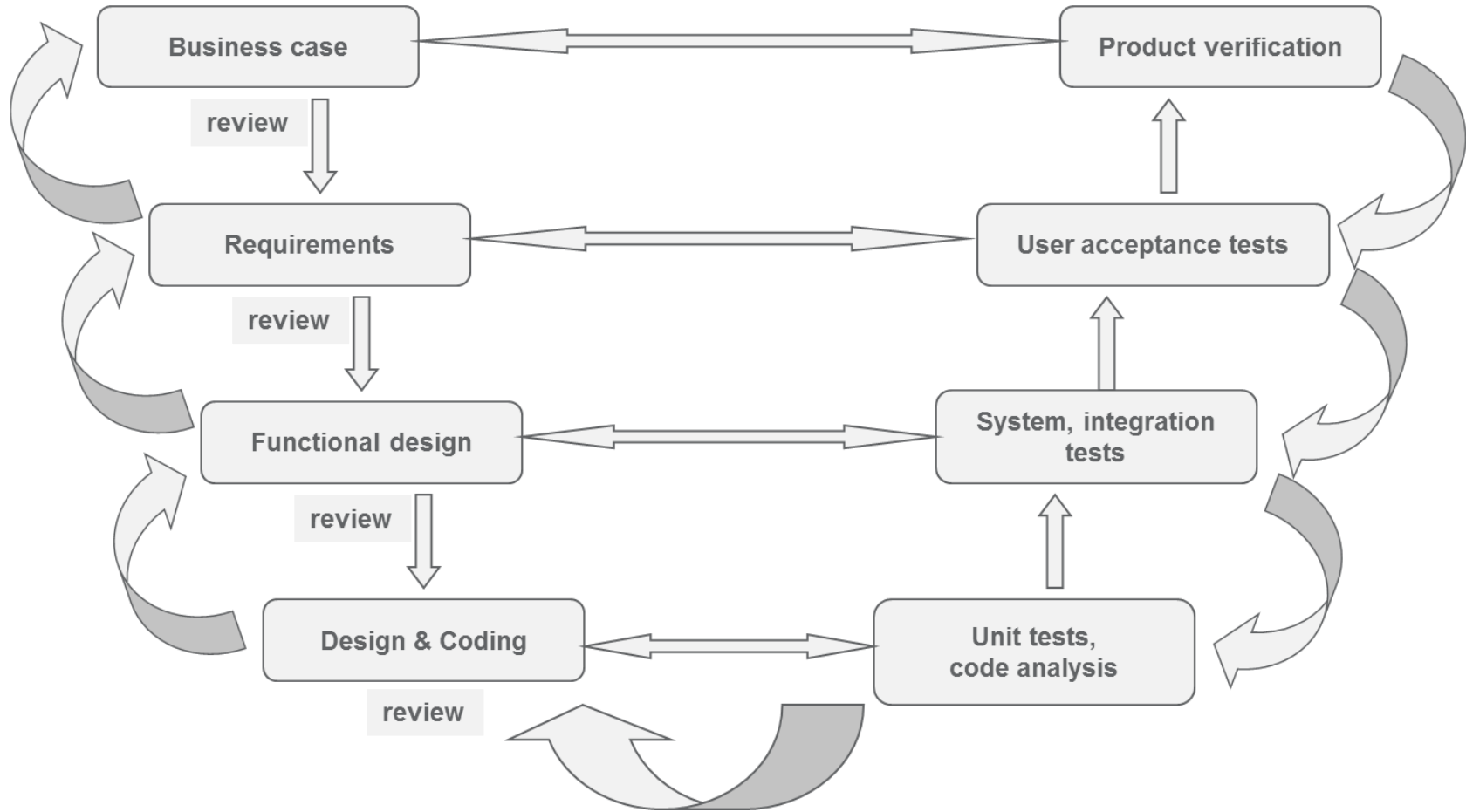


Iterations

Activities & Representative Amounts



# Načasování testů aneb „V – model“



# Testování vs zajištění/kontrola kvality

- QA != QC (Quality Assurance není Quality Control)
  - QA se zabývá procesy, QC výsledným produktem
    - QA se snaží o prevenci vzniku chyb, zatímco QC se snaží chyby nalézt.
  - V praxi se však termín často QA používá i pro QC
- Jaký je vztah testování ke QA / QC?
  - Testování je podmnožinou
- Příklady dalších aktivit zajišťování/kontroly kvality:
  - Revize (například specifikace, kódu, designu)
    - Merge requesty
  - Statická analýza kódu
  - Continuous integration

# Základní pojmy

## Test plan (plán testů)

- Definiuje strategii testů, vždy obsahuje Test coverage (**co** testovat, co netestovat), Test methods & tools (**jak** testovat a pomocí jakých nástrojů), Test responsibilities (**odpovědnosti**)

## Test case (testovací případ)

- Množina podmínek, za kterých tester určí, zda aplikace či systém funguje korektně či nikoliv

## Test script

- Množina instrukcí (kroků), které budou provedeny na testovaném systému s cílem zjistit, zda systém funguje, jak je očekáváno

## Test data (testovací data)

- Data speciálně identifikovaná pro využití v rámci testovacího případu

## Test report (výsledky testu)

- Výsledek jednoho či více testů obsahující minimálně identifikaci testu a jednoznačný výsledek společně s komentářem, je-li třeba

# Základní principy

- Kompletní testování není možné
- Práce testerů je kreativní a náročná
- Testování je „řízeno“ riziky
- Analýza, plánování a návrh jsou důležité
- Motivace je důležitá
- Čas a zdroje jsou důležité
- Časování přípravy testů hraje velkou roli
- Měření a sledování „pokrytí“ je důležité

# Typologie testů

## ○ Tři různé dimenze

- co testujeme za konfigurační jednotku
- jaký aspekt konfigurační jednotky testujeme
- s jakým cílem testujeme

- Unit testy
- Integrační testy
- Systémové testy
- Akceptační testy
  - Uživatelské
  - Operační
  - ...

- Regresní testy
- Kvalifikační testy
- ...

- Funkční testy
- Výkonové testy
- Bezpečnostní testy
- Testy dostupnosti
- Testy spolehlivosti
- ...







Začínáme testovat

# Základy základů

## Pozitivní vs. Negativní testy

- funguje, co fungovat má
- nefunguje co fungovat nemá
- neomezovat se na „přípustné“ hodnoty, operace, ...
- vždy zkoušet, jak se SW chová v případě „nepřípustných“ hodnot, operací, ...

### Black box

- Testujeme oproti „rozhraní“
- Nezajímá nás implementace
- Robustnější
  - není nutné často upravovat

vs.

### White box

- Strukturální testy
- Přihlížíme k implementaci
- Křehčí
  - změna implementace je rozbije
- „path“ testing

## Boundary and Equivalence Analysis

- Testů je příliš mnoho → **Partitioning** (rozdělení testů do **tříd ekvivalence**)
- Provádění více testů ze stejné skupiny je **redundantní**
- Volba **nejvhodnějšího reprezentanta** skupiny - ten s max. pravděpodobností odhalení chyby
- **Hraniční testovací případy** jsou obvykle velmi mocné a identifikují často chyby

# Testovací techniky

## Testovací techniky / paradigmatata

- Definuje typy testů, které jsou relevantní a zajímavé
  - Vytváří určitý způsob myšlení a přístup k testování
  - Implicitně určuje limity co je relevantní, zajímavé nebo možné
- Existuje velké množství technik, cca 150
- Překrývají se

## Jak je využíváme ke tvorbě testů ?

- Analýza situace
- Modelování testovacího prostoru
- Volba pokrytí
- Konfigurace testovacího systému
- Provoz testovacího systému
- Pozorování testovacího systému
- Zhodnocení výsledků testu

Testovací technika  
je **recept** pro  
provádění těchto  
činností s cílem  
objevit něco,  
co stojí za reporting.

# Výběr vhodné techniky

Záleží na několika aspektech

- Požadavky na testy
- Atributy testů
- Kontext vývoje
  - Elementy produktu
  - Kritéria kvality
  - Rizika
  - Omezení projektu

## Příklad

- Funkce stejně důležité
- Chyby stejně závažné
- **Která varianta je lepší ?**

Varianta I  
Varianta II

	Nalezeno před releasem
Funkce A	100
Funkce B	0
Funkce C	0
Funkce D	0
Celkem	100
Funkce A	50
Funkce B	8
Funkce C	8
Funkce D	8
Celkem	74

# Výběr vhodné techniky

	Nalezeno před releasem	Nalezeno později	Celkem	
Varianta I	Funkce A	100	0	100
	Funkce B	0	16	16
	Funkce C	0	16	16
	Funkce D	0	16	16
	Celkem	100	48	148
Varianta II	Funkce A	50	50	100
	Funkce B	8	8	16
	Funkce C	8	8	16
	Funkce D	8	8	16
	Celkem	74	74	148

# Jak vybrat vhodnou techniku

## Atributy testů

- **Power** – vysoká pravděpodobnost nalezení problému, pokud existuje
- **Valid** – odhalí skutečné chyby
- **Value** – odhalí chyby důležité pro uživatele
- **Credible** – odpovídá očekávanému chování uživatele
- **Representative** – odpovídá tomu, čeho si uživatel nejpravděpodobněji všimne
- **Non-redundant** – reprezentuje skupinu testů, které se zaměřují na stejné riziko
- **Motivating** – „klient“ bude chtít chyby nalezené testem opravit
- **Performable** – proveditelný v souladu s návrhem
- **Maintainable** – udržovatelný při změnách systému
- **Repeatable** – snadno a levně znovupoužitelný
- **Pop** (Karl Popper) – odhalí věci týkající se základních či kritických předpokladů
- **Coverage** – vyzkouší systém způsobem, kterým to nečiní jiné testy
- **Easy to evaluate** – snadné a jasné vyhodnocení
- **Appropriately complex** – dostatečná komplexnost
- **Accountable** – obhajitelnost, prokazatelnost testu
- **Supports troubleshooting** – poskytuje užitečné informace k ladění nalezených problémů
- **Cost** – přímé náklady, čas a pracnost
- **Opportunity cost** – náklady, které se ušetří provedením testu

# Dominantní „techniky“

- Function
- Specification-based
- Domain
- Risk-based
- Scenario
- Stress
- User
- High volume automated
- Exploratory
- Regression

Regresní testování není technika sama o sobě, jde o využití testů vytvořených dle jiných technik, zde explicitně vytaženo pro svou důležitost ...

# Plánování testů



# Plánování testů

## Jak neuspět

- Zapomenout, že testují lidé
- Předstírat, že testeři jsou odpovědní za kvalitu, nikoli management
- Diktovat datum spuštění bez ohledu na reálná omezení projektu
- Hodnotit testerů podle počtu nalezených chyb
- Nedostatek vzdělávání pro testery
- Oddělit vývoj a testování

## Plán testů

- Kolik času vyhradit na testování ?
- Co všechno má být v plánu testů ?
  - **Test coverage** (co testovat)
  - **Test methods and tools** (jak testovat)
  - **Test responsibilities** (odpovědnosti)

# Návrh testovacích případů

# Návrh testovacích případů

## Jak neuspět

- Malá diverzita použitých technik
  - Pouze specification based testing
  - Pouze function testing
- Příliš detailní testovací skripty
  - Malá volnost pro kreativitu testera
  - Malý prostor pro „náhodu“
  - Obtížná udržovatelnost
- Exploratory testing bez patřičného vzdělávání
- Oddělení návrhu a provádění testů
- Ignorování existujících rizik

## Jak má vypadat testovací případ ?

- Viz techniky testování
- Viz atributy testu
- Míra detailu
- Testovací data
- Standardní struktura

# Provádění a vyhodnocení testů

# Provádění testů

## Kdy začít testovat ?

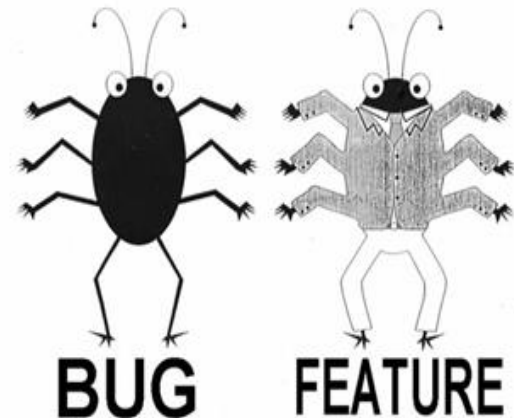
- Plánovaný harmonogram vs. realita
  - zpoždění dodavatele
  - čekat nebo začít dříve ?
  
- Dobré časování je zásadní
  - příliš **pozdě** → problém se splněním termínů, málo času na testování
  - příliš **brzo** → nestabilní SW, zbytečně vynaložený čas a práce testerů



# Trouble ticketing, bug reporting

## Základní pravidla

- Evidence **všech** nalezených **issues**
- Jediné místo pravdy
- Nejde jen o to nahlásit issue, důležité je udělat to tak, aby bylo možné jej
  - nasimulovat a
  - **opravit**
- Trouble ticketing → Bug tracking ?



# Perličky – příklady nahlášených chyb

- *Nelze se přihlásit. Na klávesnici nelze zadat znak €*
- *Zařízení MGHA12 má ožuzlanou tužku*
- *Nepřišel exot*
  - *Správná formulace by byla „OK4765 13.8. nepřišel EXOT“.*
- *Na ODP se špatně aplikuje pro rata*
- *Vždycky když jsem před hangárem F, tak to posílá blbé časy. Příjem signálu dobrý*
- *Komentáře: vybrat dokument, nezadat komentář – uložit. Validáční hláška: "Text komentáře není vyplnen"*

# Reportování výsledků testů

- Standardizovaný test report
- Celkové zhodnocení testovaného SW
- Dopad nedostatků na projekt, systém, ...
- Detailní výsledky
  - Nalezené problémy
  - Odchytky od testovacích případů
- Log testů (provedené testy, průběh testů, ...)



# Řízení průběhu testů

# Řízení průběhu testů

- Klasické aktivity jako v případě Project managementu
  - Alokace zdrojů
  - Dynamické přidělování a plánování práce
  - Reakce na problémy
  - Zlepšování procesu testování
  - Snaha optimalizovat
  - ...
- ➔ Musíme vědět „**jak na tom jsme**“ !

*„Kolik testování jste na projektu už udělali?“*

*„Jak a podle čeho měřit rozsah testování?“*

*„Co je to rozsah testování ?“*

# Řízení průběhu testů

## Co je to rozsah testování ?

- Typicky odpovědi založené na
  - **Produkt:** „Otestovali jsme 70% řádek kódu“
  - **Plán:** „Provedli jsme 65% testovacích případů“
  - **Výsledky:** „Našli jsme 753 chyb“
  - **Pracnost:** „Pracovali jsme 3 měsíce 60 hodin týdně, provedli jsme 8956 testů“
  - **Kvalita testování:** „Beta testeři našli 28 chyb, které nám unikly, naše regresní testy se zdají neefektivní“
  - **Rizika:** „Dostáváme spoustu stížností od Beta testerů, stále máme otevřených přes 500 problémů, produkt nebude do 3 dnů připraven ke spuštění“
  - **Projektová historie:** „V tento moment jsme na předchozích projektech měli 12% nalezených problémů stále otevřených, stejné by to mělo být i teď“

## Měření rozsahu testování

- Žádná metrika není dokonalá
- Řešením z praxe je ... ?
  - ... kombinace více různých metrik ...
  - pokrytí, pracnost, výsledky, rizika, potíže, ...

# Automatizace v kontextu testování

# Automatizace exekuce testů

- Snaha automatizovat testy již mnoho desítek let
- Proč ?
  - Opakovatelnost a konzistence testů → stejné vstupy a podmínky nezávisle na počtu opakování, odpadá problém s motivací lidí k opakování stejných testů
  - Praktická znovupoužitelnost testů → lze opakovat stejný test v různých prostředích, v různých konfiguracích, s mírně modifikovanými vstupními daty, ... a znovuspuštění testu je levné
  - Praktické baseline testy → automatizace umožňuje spustit velmi „hutnou“ sadu testů, umožňují efektivně provádět regresní testování

# Automatizace regresních testů

- Velice častý scénář
- Typický průběh automatizace
  - Vytvořit testovací případ
  - Manuálně jej spustit a ověřit výstup
  - V případě selhání nahlásit chybu
  - V případě úspěchu „uložit“ výsledek
  - Opakovaně spouštět test a výsledky porovnávat s uloženými, hlásit chybové situace
  - Udržovat automatický test

## Je to skutečně automatizace ?

- Analýza programu
- Design testu
- První spuštění testu
- Uložení výsledků
- Dokumentace testu
- Znovuspuštění testu
- Vyhodnocení výsledků
- Údržba testu

Co z toho vlastně  
dělá stroj ?



# Automatizace regresních testů

## Ne pro automatizaci ...

- Tvorba testovacích případů je drahá
- Vyžaduje velmi technicky zkušené členy týmu
- Vyžaduje dobře definované a stabilní rozhraní
- Vyplácí se pozdě (výhody automatizace v release N se vrací až v release N+1)
- Regresní testy mají často menší Power než nové testy
- ...

## Kdy může mít smysl ? Vždy otázka ROI !

- Smoke testing (**Continuous Integration**)
- Configuration testing (HW SW compatibility)
- **Variace** (viz. debata o Stochastic testing)
- Stress testing
- Load testing
- **Příprava testovacího prostředí** (data, ...)
- ...

# Nástroje pro automatizaci testů

## Unit a integrační testy

- [JUnit](#), [TestNG](#), [jMock](#), [EasyMock](#), [DbUnit](#), ...

## Statická analýza kódu

- [Findbugs](#), [PMD](#), [JDepend](#), FoxCop, ...

## Funkční testy – tlustý klient

- [Selenium](#), HP QuickTest, IBM Functional Tester, ...

## Funkční testy – tenký klient

- HP QuickTest, IBM Functional Tester, White, Autolt, ...

## Výkonové testy

- [JMeter](#), [DieSELtest](#), [QALoad](#), ...

## Komplexní řešení

- HP Test Suite, IBM/Rational Test Suite, ...

## Příprava testovacího prostředí

- IBM Optim, Grid-Tools DataMaker, Oracle Datamasking, ...







Goodies

# MATERIÁLY SWENG

## TESTING

### ČLÁNKY

- [What Is Software Testing? And Why Is It So Hard?](#) – absolutní klasika, jejímž autorem je guru v oblasti testování, James A. Whittaker, který řeší problematiku testování pro společnosti jako Google nebo Microsoft
- [What Is a Good Test Case](#) – velmi zdařilý (a krátký) článek, který srozumitelně a současně nezjednodušeně popisuje problematiku vytváření testovacích případů
- [Little Book of Testing – Volume I](#) – jasná a srozumitelná, obsahuje mnoho nahuštěných informací a dobrých rad pro každého, kdo přichází s testováním do kontaktu
- [Little Book of Testing – Volume II](#) – pokračování prvního dílu, obsahuje další pokročilá témata
- [Software Test Planning and Management Guide](#) – průvodce tvorbou a managementem plánu testů vytvořený ve Space and Naval Warfare Systems Center San Diego

### CHECKLISTS

- [CxCheck\\_TestPlan.pdf](#) – checklist firmy Construx vhodný pro vytváření plánu testů

### TEMPLATES

- [CxTemp\\_TestPlan.doc](#) – template firmy Construx pro plán testů
- [IEEE\\_TestPlanTemplate.pdf](#)
- [MIL-STD-498\\_SwTestDescriptionTemplate.doc](#)
- [MIL-STD-498\\_SwTestPlanTemplate.doc](#)
- [MIL-STD-498\\_SwTestReportTemplate.doc](#)
- [SPAWAR\\_SwTestPlanTemplate.doc](#)
- [SPAWAR\\_SwTestReportTemplate.doc](#)

Všechny odkazované materiály jsou poskytnuty výhradně za účelem výuky softwarového inženýrství.

## OBLASTI SWENG

› [Project and process](#)

› [Project management](#)

› [Documentation](#)

› [Quality assurance](#)

› [Configuration management](#)

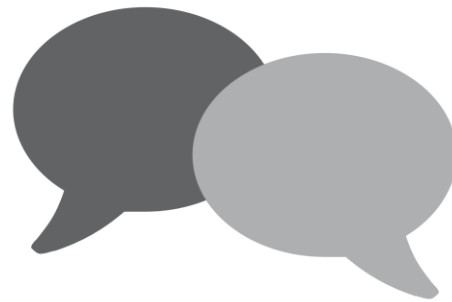
› **[Testing](#)**

› [Construction](#)

› [Architecture and design](#)

› [Requirements](#)

› [Obecné, přehledové](#)



# Diskuze

# Děkujeme za pozornost

PROFINIT

Profinit, s.r.o.  
Tychonova 2, 160 00 Praha 6



Telefon  
+ 420 224 316 016



Web  
[www.profinit.eu](http://www.profinit.eu)



LinkedIn  
[linkedin.com/company/profinit](https://linkedin.com/company/profinit)



Twitter  
[twitter.com/Profinit\\_EU](https://twitter.com/Profinit_EU)



# Backup slides – dominantní techniky

# Dominantní techniky

## Function testing

- Test každé funkce / feature v izolaci
- Použití „opatrných“ (middle-of-the-road) hodnot
- Neočekáváme selhání
- Později „přitvrdíme“
- *Highly credible a Easy to evaluate* testy
- Opomíjí interakce a průzkum přínosů programu

## Specification-based testing

- Test SW proti každému tvrzení v referenční dokumentaci (specifikace, uživatelský manuál, ...)
- Závisí na kvalitě referenční dokumentace
  - Nutná revize
  - **Neúplnost, nejednoznačnost, netestovatelnost**
- *Traceability matrix* (pokrytí „tvrzení“ testy)
- *Highly significant (motivating)* testy
- Hledá špatně specifikované oblasti

# Dominantní techniky

## Risk-based testing

- Představit si možné způsoby, jak může program selhat, a navrhnout jeden či více testů, které ověří, zda skutečně daným způsobem selže.
- Snaha najít „**velké problémy**“ co nejdřív
- **Optimalizace** priorit dle míry rizika
- „Kompletní“ množina risk-based testů založena na úplném seznamu rizik (věcí, které mohou jít špatně)
- Nebezpečí **špatné identifikace** rizik
- *Highly credible, highly motivating* testy
- Potenciál mít *high information value*

# Dominantní techniky

## Risk-based testing – kde hledat problémy

- Nesplnění některých atributů kvality
- Nové funkce, technologie
- Věci dělané na poslední chvíli
- Unavení, problémoví programátoři
- Nejasnosti (ve specifikaci, ...)
- Komplexní funkce
- Historicky chybové funkce
- ...
  
- Hlášené problémy ostatních
  - <http://www.bugnet.com>
  - <http://www.cnet.com>
  - odkazy na <http://www.winfiles.com>
  - některé mailing listy a další zdroje ...



# Dominantní techniky

## Scenario testing

- Testy jako komplexní „příběhy“, které odpovídají způsobu použití programu v reálných situacích
- Nutno dodržet následující atributy
  - *Complex* (mnoho funkcí)
  - *Credible, Motivating* (odpovídá reálnému použití)
  - *Easy to evaluate* (žádná nejasnost, zda test proběhl či selhal)
- *High power testy*
- Jedná chybná funkce zhatí vše
- Nebezpečí nedostatečného pokrytí
- Inspirace
  - Dle produktů konkurence
  - Způsoby chování zákazníka
- Varianta „*harsher*“ testy (*killer soap opera* 😊)
  - Běžný scenario test, pouze jiná sekvence či ne tak běžná data
  - Netypické chování uživatele

# Dominantní techniky

## Domain testing

- Testy proměnných (vstupy, konfigur. hodnoty, ...)
- Proměnné v izolaci i ve skupinách
- Všechny přípustné i nepřípustné hodnoty
- Partitioning, třídy ekvivalence, „nejlepší zástupce“
- Pozor na chyby mimo hraniční případy
- *High power, Lot of Information value* testy
- Nižší *Credible a Motivating* (corner cases)

## Regression testing

- Vytváření testů s důrazem na jejich pravidelné opakování po provedení změn v programu
- Libovolný test lze použít jako regresní
- Zpočátku *Power* a *Credible* testy, jejich síla klesá s časem a počtem jejich opakování (pokud se nedějí v systému velké změny)
- Je nutné se naučit navrhovat testovací případy s důrazem na znovupoužitelnost

# Dominantní techniky

## Stress testing

- Několik definic
  - Zatížit program extrémní aktivitou a sledovat, zda selže
  - Testování za hranicemi specifikovaných požadavků na komponentu či program s cílem způsobit selhání systému
  - Zahnání aplikace nestandardními podmínkami do stavu selhání s cílem sledovat **JAK** program selže a jaká slabost se tímto odhalí
- *High power* testy
- Někdy ne tolik *Credible* a *Motivating* testy (netypické pro uživatele)
- Problémy s **diagnostikou** v případě selhání

## High volume automated testing

- Random / Stochastic testing
- Strukturu testu navrhuje člověk, jednotlivé testovací případy vyvíjí, spouští a interpretuje stroj, který následně upozorňuje na selhání
- Nutná maximální (kompletní) míra automatizace
- Testy někdy slabé, málo *Credible* a *Motivating*, není to „řemeslná práce“
- Nutno zabudovat troubleshooting

# Dominantní techniky

## User testing

- Testování, které dělají **skuteční** uživatele, ne testeři předstírající uživatele
- Navrhovat testy mohou testeři
- Libovolný test lze použít jako User test
- Důležité je nechat uživateli dostatek „prostoru“
- *Credible a Motivating* testy

## Exploratory testing

- Libovolné testování takové, že
  - tester **aktivně** kontroluje návrh testu **během** jejich **provádění** a
  - využívá informace nabyté během testování k návrhu nových a lepších testů
- Očekávané výsledky ani konkrétní podoba testů není definovaná, záleží na uvážení testera
- Používání mnoha stylů, podle toho, který je zrovna nejvhodnější pro splnění cíle
- Nutný velmi **zkušený tester** se znalostí produktu, domény, test. technik, ...
- Vhodné pro **párové** testování