

Automatizace testování

Radek Mařík

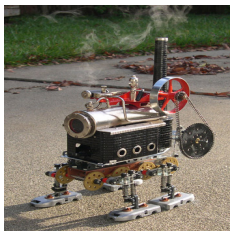
CA CZ, s.r.o.

September 14, 2007



- 1 Motivace
 - Stav a cíle
- 2 Pojem automatizace
 - Obecná definice a její interpretace
- 3 Automatizace testování softwaru
 - Proč (ne)automatizovat?
 - Praktická řešení
- 4 Automatizace návrhu testů
 - Základní principy
 - Náhodné testování - opice

Proces vývoje a proces testování



Trendy

- Komplexita softwaru překotně stoupá.
- Jednoduchá modifikace implementace software může způsobit velké množství změn v testovacích skriptech.

- Nástroje

- Vývojáři používají pokročilé techniky jako průvodce, CASE nástroje.
- Testeři kódují každou řádku manuálně.

- Použití abstrakce

- Software používá abstraktní metody, aby pokryl velké množství případů.
- Testware se musí implementovat každý případ zvlášť.

Požadované základní vlastnosti procesu testování

Žádá se, stěžuje se na, diskutuje se, ...

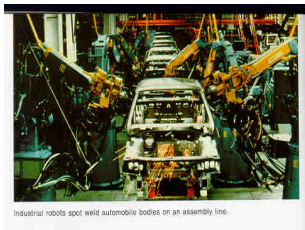
- **Opakované použití:** testovací metodika by neměla být vyvíjena pouze pro jediný projekt.
- **Flexibilita:** vyjádření nových konceptů, návrhových šablon.
- **Adaptivita:** malé modifikace v implementaci softwaru by měly být pokryty automatizovaně.
- **Komplexita:** pokrytí dostatečné části testovacích případů je často za možnostmi manuální přípravy.

Požadované odvozené vlastnosti procesu testování

Potřebné pro řízení projektů

- **Údržba:** potřebné úsilí je
 - nepřímo úměrné flexibilitě a adaptivitě,
 - přímo úměrné komplexitě testovaného produktu.
- **Prezentace stavu:** dokumentace pravidelně obnovovaná, např. WWW stránky.
- **Nástroje:** integrovaná řešení adresující výše uvedené položky.
- **Cena/čas:** efektivnost vyjádřená pomocí výše uvedených položek.
- **Proveditelnost:** trh/cena/čas/zdroje/kvalita efektivnost.
- **Nepřetržitý běh:** rychlá odezva, několik fází (zahořovací, . . . , regresní, dokumentace pravidelně obnovovaná, např. WWW stránky.

Co je to automatizace?



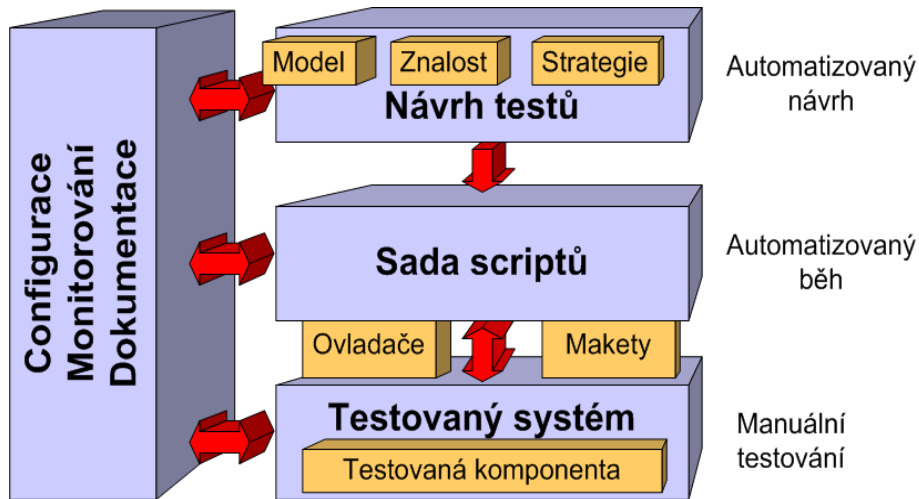
Automatizace

je proces, při kterém se přiřazují strojům či systémům aktivity, které vykonávali ještě donedávna lidé.

Základní stavební bloky

- formální modely,
- zpětná vazba,
- programování.

Úrovně automatizace



Automatizace testování ?

Výhody

- 1 Běh regresních testů na nové verzi programu.
- 2 Častější testování.
- 3 Provedení testu, který by jinak bylo obtížné provést.
- 4 Lepší využití prostředků.
- 5 Konzistence opakovatelnosti testů.
- 6 Vícenásobné použití testů.
- 7 Zkrácení doby uvedení na trh.

Problémy

- 1 Nereálná očekávání.
- 2 Slabá testovací praxe.
- 3 Očekávání, že automatizovaný test nalezne mnoho nových defektů.
- 4 Údržba automatizovaných testů.

Porovnání postupů

Vlastnost	Manuální testování	Automatizovaný běh	Automatizovaný návrh
Cena přípravy testovací sady	Nízká (omezený rozsah)	Vyšší	Velmi vysoká Nízká (existující řešení)
Kombinatorické pokrytí	Neschopné	Velmi omezené	Řízené
Flexibilita a adaptivita	Vysoká (lidé)	Zanedbatelná (přejmenování)	Vysoká
Cena běhu	Vysoká	Nízká	Nízká
Vstupy	Vágní	Vágní	Modely softwaru
Detekční schopnost	Vysoká	Nízká	Střední

Charakteristiky manuálního testování



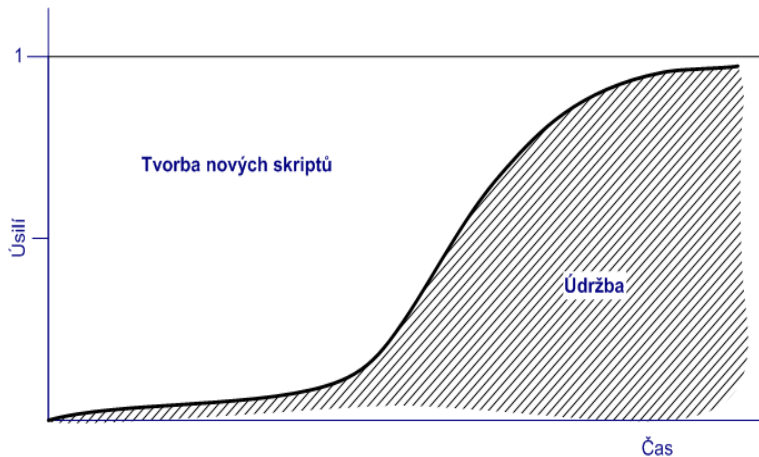
- Vysoce flexibilní a adaptivní (lidé)
- Vysoká detekční rychlost defektů
 - Lidé jsou schopni ovládat software a vyhledávat chyby i za situace, kdy software je ve velmi špatném stavu.
 - Použití objevného testování (exploratory testing) během počátečních vývojových fází.

Charakteristiky automatizovaného běhu

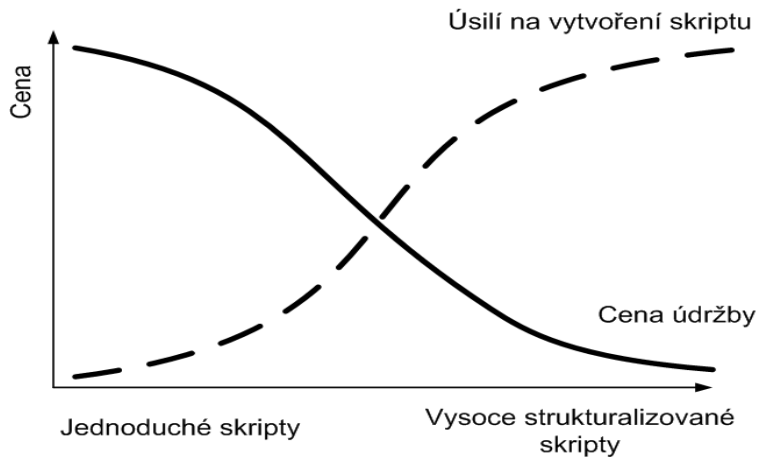


- Nízká cena opakovaných běhů.
- Obtížná údržba testovacích skriptů:
 - Strukturální změny znamenají plnohodnotnou manuálně vedenou reimplementaci skriptů.
 - Současné komerční nástroje umožňují pouze přejmenovávání položek s poloautomatizovanou detekcí.


Problém údržby



Úsilí vývoje



Charakteristiky automatizovaného návrhu

- Počáteční investice do přípravy technologie
- V současnosti, **vysoká cena přípravy** (řada technologií není ještě dostatečně zmapována).
- V budoucnosti, **nízká cena přípravy** vzhledem k možnostem opakovaného použití jejích komponent.
- Schopnost adresovat **kombinatorické problémy** (vážení čas/prostor/risk)
- **Střední detekční rychlost defektů** v rámci celého vývojového procesu.
- **Flexibilita a adaptivita** řízená **testovacími modely testovaného softwaru**.
- Modely testovaného softwaru pomáhají v **porozumění** softwaru.
- **Nízká cená údržby** testovacích skriptů (např. opakované generování).
- Prakticky **zanedbatelná cena** údržby a běhů regresního testování. 

Testování kontra Verifikace

Testování softwaru

- Neformální/formální oraculus pro validaci skutečných výsledků.
- **Řízené vzorkování** chování softwaru za účelem snížení **pravděpodobnosti** selhání softwaru či **míry nespokojenosti** zákazníka.

Verifikace softwaru

- Založená na formálních modelech softwaru.
- Matematický **důkaz**, že model softwaru je správný.
- Pouze omezená množina použitelných technologií:
 - Vnořený software založený na konečných automatech.
 - Verifikace protokolů.

Automatizace provedení testů - všeobecně

- propagována komerčními výrobci,
- používána v praxi, zvláště pro regresní testy,
- **velmi citlivá na změny specifikací a implementace softwaru,**
 - testování řízené tabulkami (funkce, hodnoty parametrů)
 - testování řízené daty (šablony skriptů, hodnoty proměnných)
 - mapy grafických rozhraní (fyzické adresy - logická jména)
- dobrá vazba na navazující podporu
 - správa logů,
 - hlášení problémů.

Automatizace provedení testů - nástroje



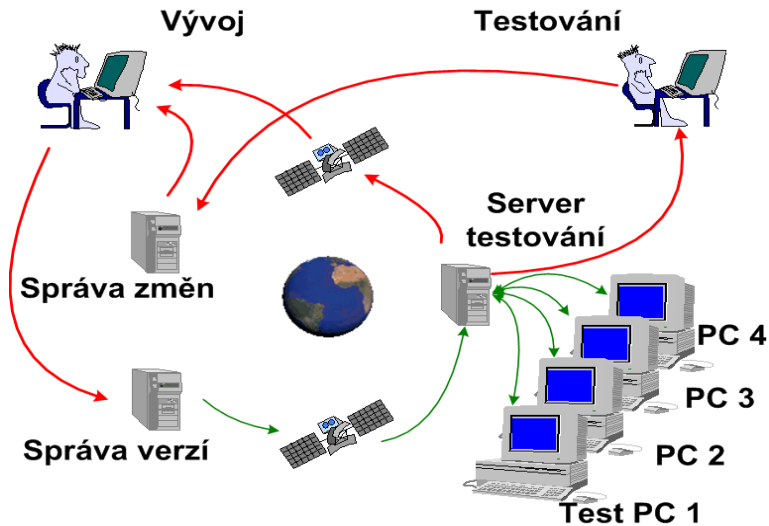
Komerční automatizační nástroje

- 2 operační fáze:
 - 1 programování pomocí simulace běhu
 - 2 automatické testování
- IBM Rational Robot,
- IBM Functional Tester,
- Mercury WinRunner,
- Compuware TestPartner.

Automatizační ovladače na úrovni komponent

- Expect, NUnit, JUnit, pyUnit, JavaTest, PyTest, ParaSoft, Rational Robot & Quality Architect
- šablony testů
- dosazování parametrů podle kombinačních tabulek

Distribuce zdrojů



Automatizace návrhu testů - pohled manažera

- Kratší perioda vývoje softwarového produktu.
- Vyšší spolehlivost
- Standardizovaná rozhraní softwaru
- Redukce ceny vývoje testwaru vzhledem k možnostem opakovaného použití.
- Možnost jasnějších formalizovaných specifikací komponent softwaru.
- Zmenšení zátěže testerů-lidí, které by jinak museli vyvinout obrovské množství testů pro složité softwarové aplikace a strádat jejich údržbou.
 - OO: dědičnost, polymorfizmus tříd kombinovaný s asociacemi,
 - GUI: "táhni a pusť" akce s různými zdroji a cíli,
 - GUI: spousta formulářů generovaných podle formulářů.

Automatizace návrhu testů - pohled testera

- Proč?
 - příprava velkého množství testů:

OO testing

- 10^2 tříd,
- 10^3 relací,
- 10^1 vlastností nebo metod,
- 10^5 a 10^6 testů.

GUI testing

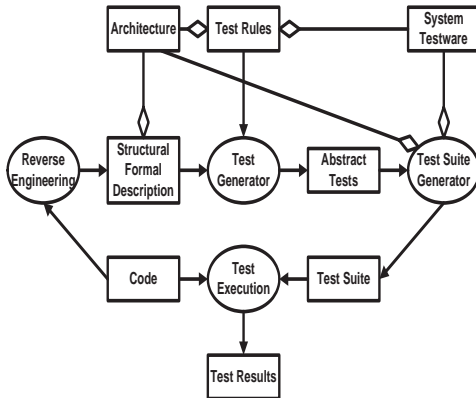
- 10^2 formulářů,
- 10^2 přechodů,
- 10^1 atributů nebo akcí,
- 10^4 a 10^6 testů.

- modifikace skriptů při změnách softwarové
 - specifikace, nebo
 - implementace (GUI).
- poměrně vysoké náklady
- technologie nedospěla do plně rozvinuté komerční fáze.

Automatizovaný návrh testů - postup

- ❶ volba testovací strategie - testovací katalog,
- ❷ určení testovacího případu,
- ❸ doplnění na testovací specifikaci:
 - ❶ vytvoření nezbytného prostředí pro testovaný objekt,
 - ❷ uvedení objektu do požadovaného stavu,
 - ❸ provedení testu,
 - ❹ uvedení testované komponenty do původního stavu,
- ❹ transformace testovací specifikace do sekvence,
- ❺ určení očekávaných výsledků,
- ❻ kompilace kódu testovacího skriptu,
- ❼ vytvoření doprovodné dokumentace.

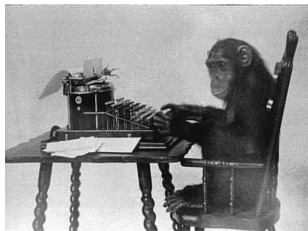
Automatizovaný návrh testů



Formální specifikace softwaru - testovací modely

- manuální vstup,
 - plná specifikace bývá náročnější než přímé kódování,
 - zatím nezbytné pro modely chování,
- analýza vhodných modelů a dat softwaru,
 - UML, IDL, SQL, XML,
 - analýza textových specifikací ve vhodném formátu,
- rekonstrukce pomocí reverzního inženýrství a validace modelu,
 - COM rozhraní vlastností objektů,
 - deterministické konečné automaty,
 - problémy se skrytými stavy,
 - problémy s kombinatorickou explozí stavů jednoduchých automatů,
- formální metody ^[Jac97, WD96],
 - aplikace logiky a jednoduché matematiky na programování,
 - zápis formálního popisu programu,
 - užívány pro modelování, návrh a verifikaci dokazováním.

Metoda pokusné opice



Myšlenka

Pokud milión opic bude po milión let náhodně "bušit" do klávesnice, mohly by statisticky nakonec napsat nějakou Shakespearovu hru či Jiráskovo Temno.

Hloupá opice

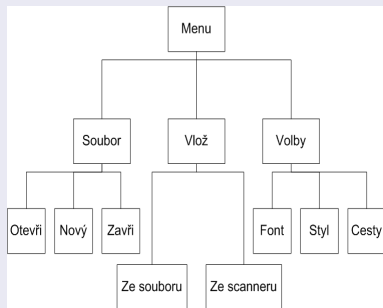
- Neví o testovaném softwaru vůbec nic.

- Jednoduchá implementace

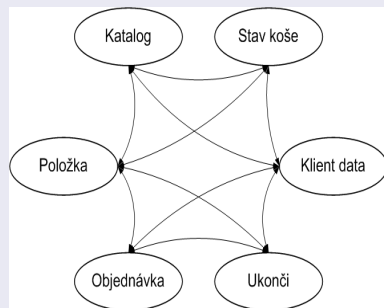
```
RANDOMIZE TIMER
FOR i=1 TO 10000
PLAY " {CLICK " + STR$(INT(RND+640)) + ". " + STR$(INT(RND*480)) + " }"
PLAY CHR$(RND*256)
NEXT i
```


Chování opic a typy aplikací

Hierarchie



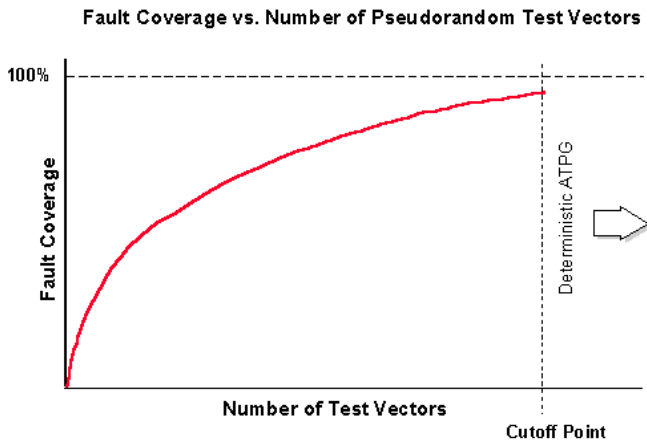
Síť



Zvýšení inteligence opice

- záznamy do protokolu,
- zaměření akce na konkrétní aplikaci,
- detekce, kde se opice nachází,
- co na tomto místě může udělat,
- kam může přejít,
- kde již byla,
- jestli to, co vidí, opravdu správné,
- přečtení (zjištění) mapy stavů a přechodů daného softwaru

Hranice metody pokusné opice



Literatura I



Jonathan Jacky.

The Way of Z: Practical Programming with Formal Methods.
Cambridge University, 1997.



Jim Woodcock and Jim Davies.

Using Z: Specification, Refinement, and Proof.
Prentice Hall, 1996.