Úvod do testování a verifikace

Radek Mařík

ČVUT FEL, K13132

20. října 2016

Obsah

Studie softwarových projektů

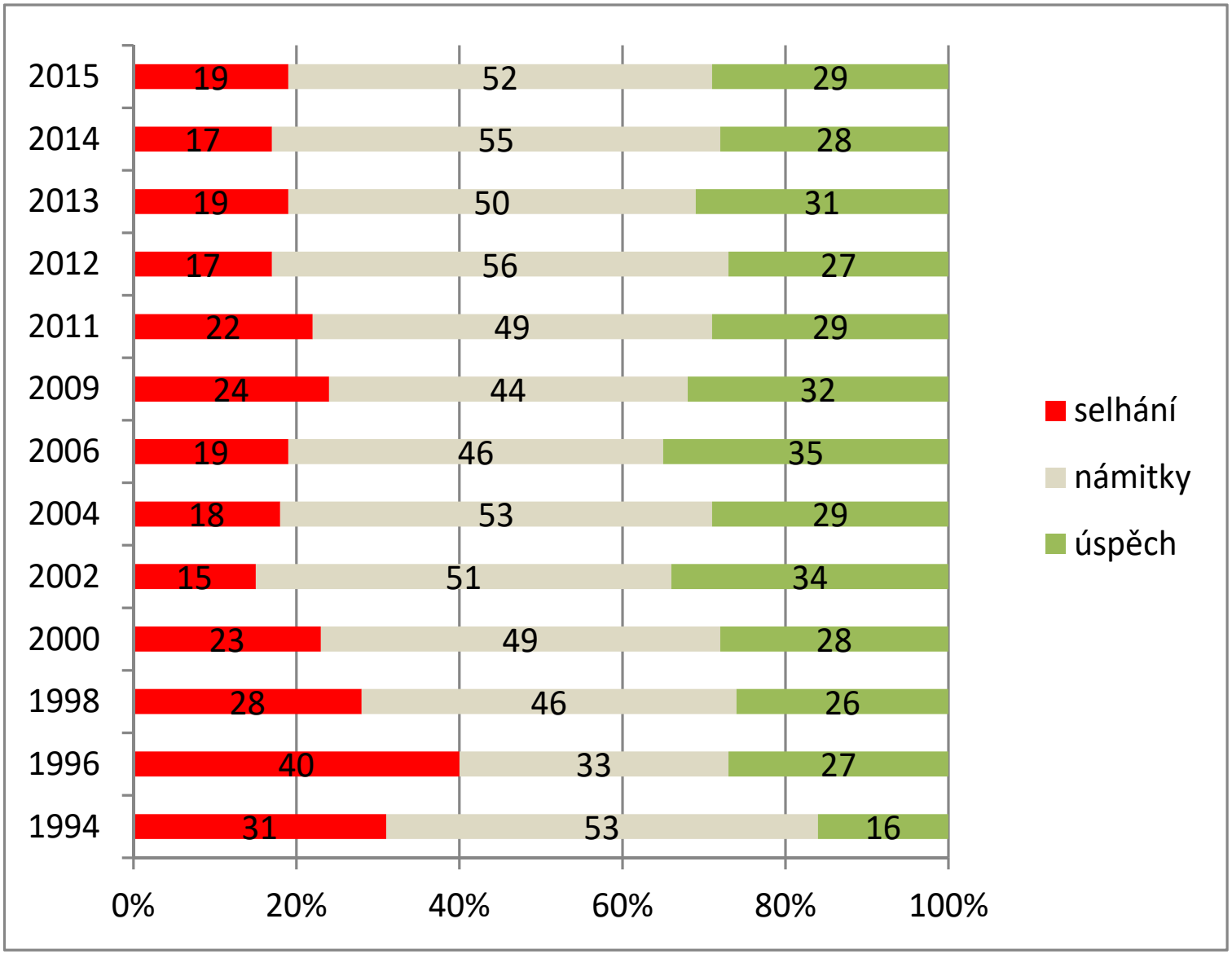
The Standish Group, 1994

- studie 8 380 projektů,
- 31% softwarových projektů přerušena,
- náklady 53% dokončených projektů se pohybují okolo 189% původních odhadů,
- z těchto 53% pouze 42% obsahuje původní sadu navrhovaných vlastností a funkcí,
- pouze 9% z těchto projektů bylo ukončeno v dohodnuté době a ceně.

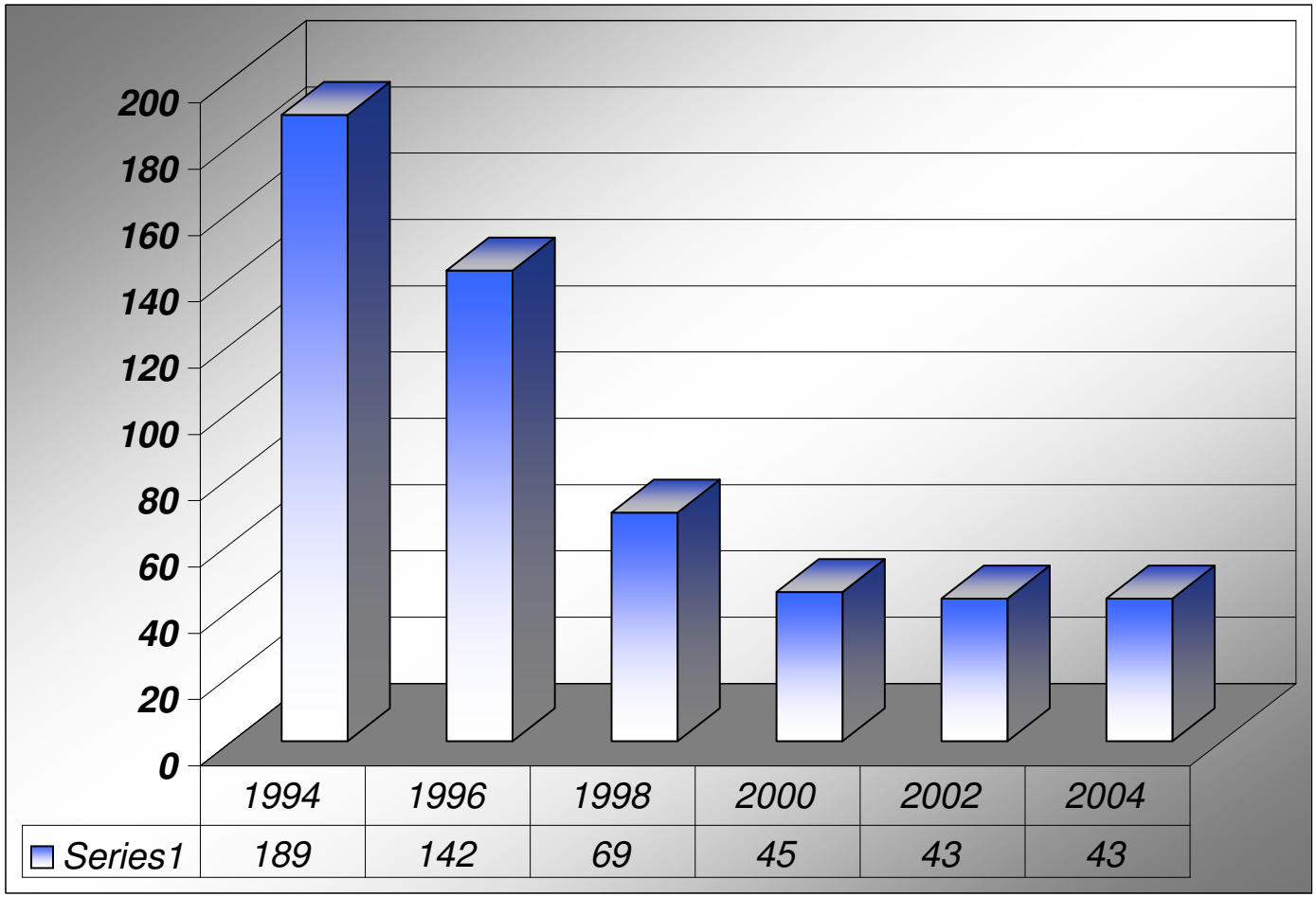
Obecně

- 5 ze 6 softwarových projektů je neúspěšných,
- 1/3 projektů je přerušena,
- *projekty předávány za dvojnásobnou cenu než dohodnuto*,
- *projekty se předávají za dvojnásobně dlouhou dobu než se plánuje*.

Vývoj úspěšnosti projektů (CHAOS)



Rozpočty projektů (CHAOS)



Agilní vs. Vodopád (CHAOS)

CHAOS RESOLUTION BY AGILE VERSUS WATERFALL

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

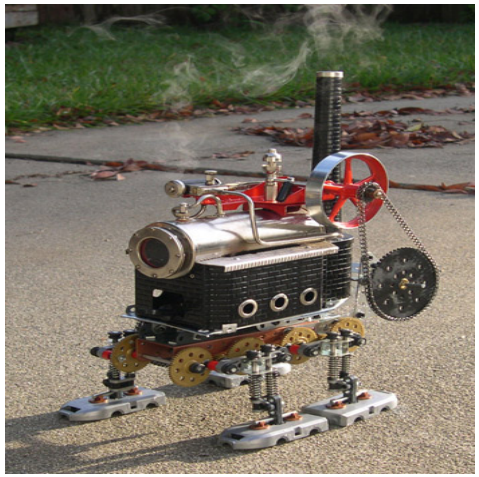
The resolution of all software projects from FY2011–2015 within the new CHAOS database, segmented by the agile process and waterfall method. The total number of software projects is over 10,000

Faktory úspěchu (CHAOS)

CHAOS FACTORS OF SUCCESS

FACTORS OF SUCCESS	POINTS	INVESTMENT
Executive Sponsorship	15	15%
Emotional Maturity	15	15%
User Involvement	15	15%
Optimization	15	15%
Skilled Resources	10	10%
Standard Architecture	8	8%
Agile Process	7	7%
Modest Execution	6	6%
Project Management Expertise	5	5%
Clear Business Objectives	4	4%

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, ...

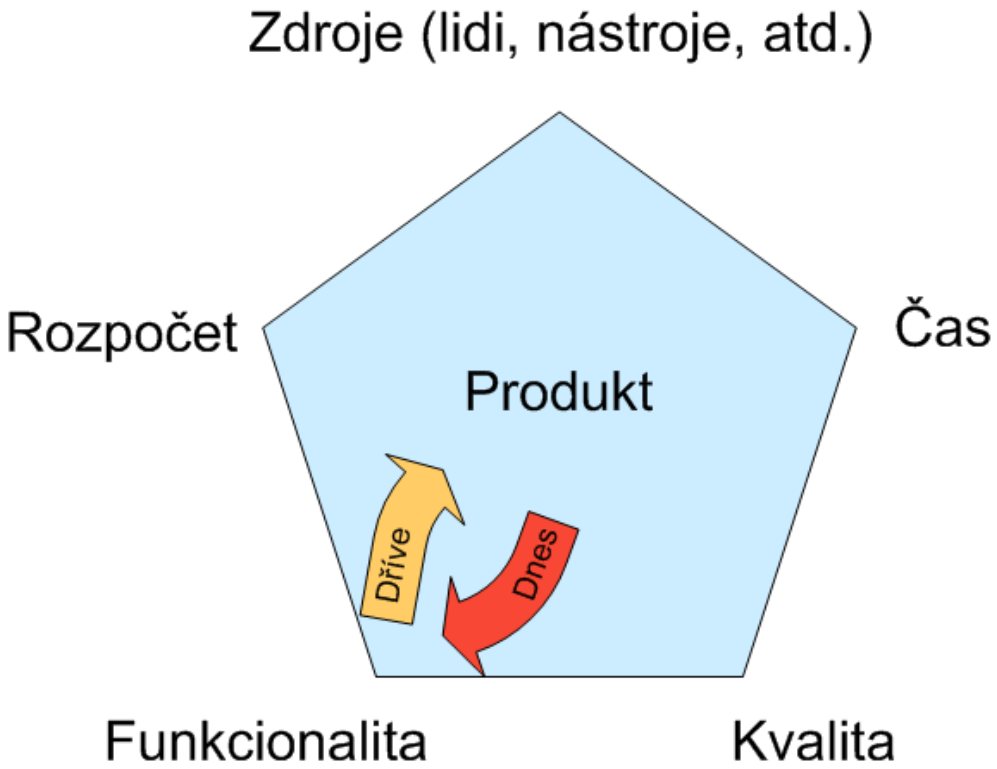
- **Opakované použití:** testovací metodika by neměla být vyvíjena pouze pro jediný projekt.
- **Flexibilita:** vyjádření nových konceptů, návrhových šablon.
- **Adaptivita:** malé modifikace v implementaci softwaru by měly být pokryty automatizovaně.
- **Komplexita:** pokrytí dostatečné části testovacích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

Požadované odvozené vlastnosti procesu testování

- **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.

Ovlivňování výsledku projektu

- Zákazníci, zadavatelé, manažéři - hodnoty libovolných **čtyř** proměnných.
- Vývojový tým - výsledná hodnota zbývající páté proměnné.



Ariane 5

Situace

- Řada motorů na tekuté a tuhé palivo nahrazena několika s větším tahem.
- 4.června, 1996, 40 s po startu ve výšce okolo 3700 m se nosič odklonil od své dráhy, rozlomil a explodoval.
- Raketa, nesené 4 satelity nepojištěny, 500 miliónů $.

Selhání nosiče Ariane 5

Status testování

- Žádost o přetestování stabilizační plošiny převzaté z Ariane 4 v podmínkách Ariane 5 byla "vetována CNES z důvodu vysokých nákladů".
- Sextant Avionique po havárii potvrdila, že by závadu svými testy detekovala.

Chyba

- softwarová vyjímka v obou Stabilizačních referenčních systémech (SRI).
- nechráněný převod z 64bitového reálného čísla na 16bitové celé číslo.
- SRI má význam pouze před zvednutím nosiče, ačkoliv je operativní jestě dalších 50 s.
- přetečení nastalo z důvodu rozdílných drah.

Shrnutí Arian 5

Typické chyby při procesu vývoje softwaru

- nedostatek času - veto na testování
- malé či chybně rozložené náklady - veto na testování,
- chybné nebo chybějící požadavky - jak dlouho by měl podsystém fungovat,
- chyby v kódu - nechráněné převody,
- opakované využití - změna specifikací,
- řada chyb vzniká striktním oddělením vývoje softwaru a jeho testování.

Radiační předávkování

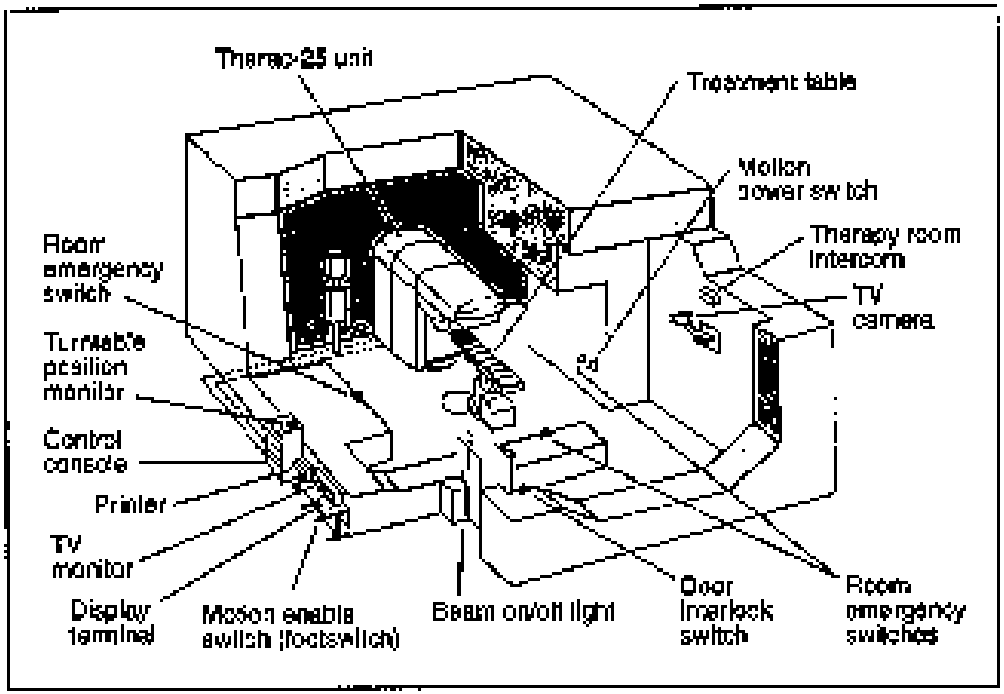


Figure 1. Typical Therac-25 facility.

Therac 25

- červen 1985 - leden 1987
- lineární urychlovač
- používaný v lékařství k ozařování rakovinných nádorů,
- povrchové tkáně ozařovány elektrony,
- pro hlubší tkáně gama záření,
- 6 incidentů přezáření, z toho 3 smrtelné,
- 20000 rad místo 86 rad,
- systém reálného času vytvořený 1 programátorem,
- neexistující formální specifikace a testovací kritéria,
- hardwarové zámky nahrazeny programovými,
- pokud byla vstupní data změněna mezi 1 až 8 s, pak zářič a polohovací stůl pracovaly v různých módech,
- k nastavování logické proměnné použita inkrementace bytové proměnné.

Shrnutí Therac 25

- uživatelské rozhranní kontra bezpečnost,
- složitý návrh
- systémové testování není dostatečné,
- chybějící specifikace,
- typicky problémy systémů:
  - paralelních (angl. parallel)
  - souběžných (angl. concurrency).

Zkušenosti z chyb

Proslulé chyby

Oběžná dráha Apollo 13: program testován za pomalu měnících se podmínek. Při velké dynamice došlo k vydělení nulou na netestované cestě.

Mariner let k Venuši: 80 miliónů $,
záměna − za + vedla k odklonu z dráhy,

Minutí Merkuru: proměnná Fortranu *DO10I*
DO 10 I=1.5
DO 10 I=1,5

Selhání rakety Patriot: během Války v zálivu v 1991 kvůli kumulativní chybě v časové synchronizaci
(ve skutečnosti: 0.34 s, 100 hodin; navrženo: 14 hodin),

F16 simulace: letadlo se překlápělo při překročení rovníku,

Návrh jaderné elektrárny: v roce 1979 muselo být 5 jaderných elektráren uzavřeno z důvodu poddimenzování potrubí, velikost vektoru počítána jako součet složek, modul byl napsán studentem na praxi.

Testování softwaru - výchozí definice

Hetzel 1973 Testování je proces určení věrohodnosti, že program či systém dělá to, co se o něj předpokládá.

Myers 1979 Testování je proces spouštění programu či systému s úmyslem nalézt chyby.

Hetzel 1983 Testování je jakákoliv aktivita s cílem vyhodnotit atribut či schopnost plnění požadovaných výsledků programem nebo systémem.

Testování softwaru - přehled definic

Testování je

- kontrola programů vzhledem ke specifikacím,
- nalézání chyb v programech,
- určení míry akceptování uživatelem,
- ujištění se o tom, že systém je připraven k používání,
- získání důvěry, že program pracuje,
- prezentace, že program běží správně,
- demonstrace toho, že program je bez chyb,
- porozumění omezení výkonnosti programu,
- učení se toho, co systém není schopen dělat,
- hodnocení schopností programu,
- verifikace dokumentace.

Testování

je měření kvality softwaru.

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.

Verifikace a Validace [Kit95, KFN93]

Verifikace

- dle definice IEEE/ANSI, je proces hodnocení systému či komponenty s cílem určit, zda produkty dané vývojové fáze splňují podmínky dané na začátku této fáze.
- Program se verifikuje vzhledem k nejblíže určujícím dokumentům nebo specifikacím. Jestliže existuje externí specifikace, pak funkční test verifikuje program vůči této specifikaci.

Validace

- dle definice IEEE/ANSI, je proces hodnocení systému či komponenty během nebo ke konci vývojového procesu s cílem určit, zda splňuje specifikované požadavky.
- Program je validován vůči publikovaným požadavkům uživatele nebo systémovým požadavkům.

Testování = verifikace + validace $\approx$ testování bílé + černé skříňky

Definice kvality

Rozsah od inženýrských specifikací na úrovni dílny až po definice na úrovni společnosti:

Webster's New World Dictionary Kvalita je fyzická či jiná charakteristika, která definuje základní podstatu věci či jednu z jejích vyznačných vlastností.

Crosby 1979 Kvalita je mírou souhlasu s požadavky.

ISO 9000 Kvalita je souhrn vlastností a charakteristik produktu či služby, která se týká schopnosti uspokojit určené nebo vyplývající potřeb.

Taguchi 1986 Kvalita je ztráta, kterou produkt způsobí společnosti po jeho dodání, způsobenou funkčními změnami a škodlivými účinky mimo těch, které vyplývají z vlastních funkcí.

Aspekty kvality

- operační podmínky - výkonnost v krátkodobém horizontu,
- spolehlivost - dlouhodobý horizont,
- pohled zákazníka,
- IKIWISI - Guaspari: "I Know It When I See It" [Kit95]

Ideální kvalita,

- kterou zákazník může očekávat, je,
- že každý produkt poskytuje cílenou výkonnost
- kdykoliv je použit,
- za všech zamýšlených operačních podmínek,
- po celou dobu jeho předpokládaného života,
- se žádnými škodlivými postranními efekty.

Kvalita softwaru

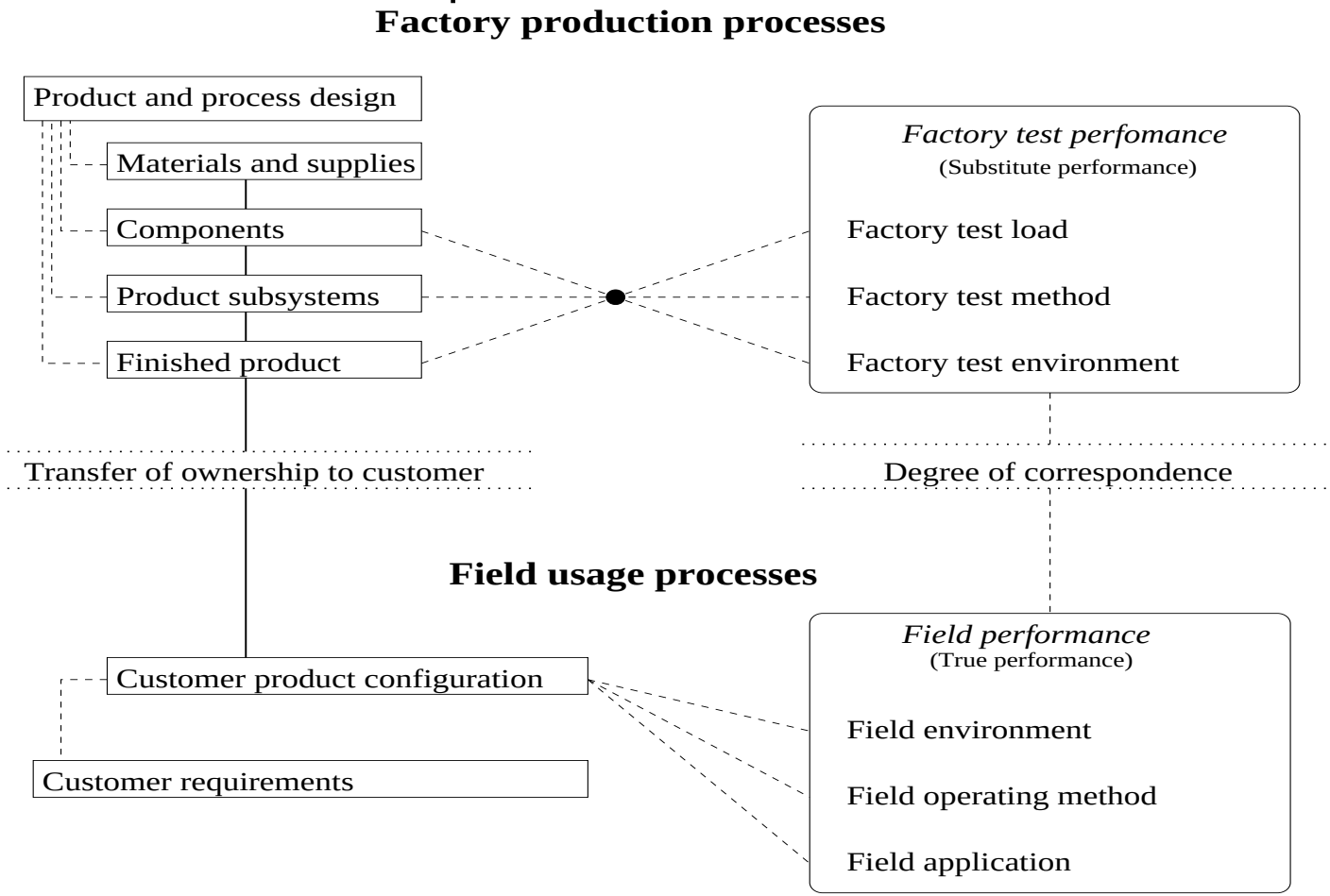
Kvalita znamená "splňovat požadavky zákazníka":

Faktory:

- *Funkčnost (externí kvalita)*
  - správnost,
  - spolehlivost,
  - použitelnost,
  - integrita.
- *Inženýrské řešení (vnitřní kvalita)*
  - efektivita,
  - testovatelnost,
  - dokumentace,
  - struktura.
- *Adaptabilita (budoucí kvalita)*
  - flexibilita,
  - opětné použití,
  - údržba.

Koncept kvality

Vztah mezi skutečnou kvalitou produktu pociťovanou zákazníkem a kvalitou měřenou na úrovni produkce:



Proaktivita a reaktivita [Kol95]

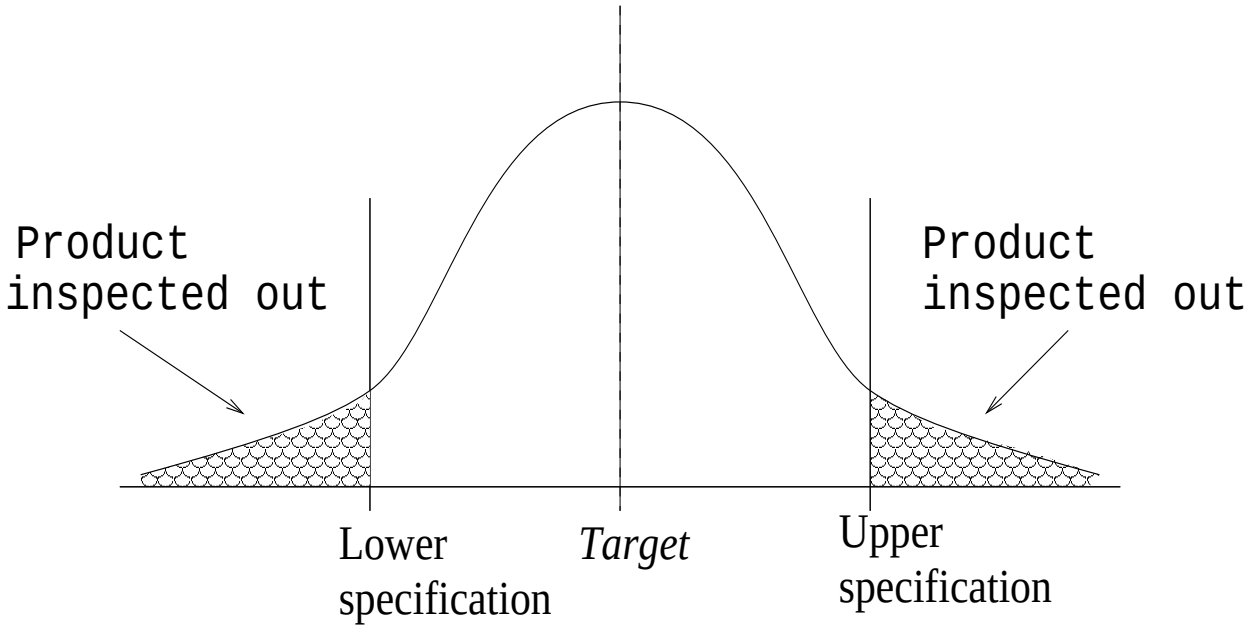
Reaktivní zabezpečení kvality

- je zaměřeno na detekování a korigování problémů, které již nastaly.
- zdůrazňuje vyhodnocování tradičních ztrát a statistické analýzy nashromážděných pozorování pro podporu akce.
- vede k omezování ztrát.

Proaktivní zabezpečení kvality

- se orientuje na prevenci,
- dává důraz na znalost příčin a následků, riskové analýzy, zkušenosti, zdůvodnění akcí,
- staví na vyšší úrovni spekulace a risku,
- vede k urychlenému vývoji,
- umožňuje vyhnutí se ztrátám.

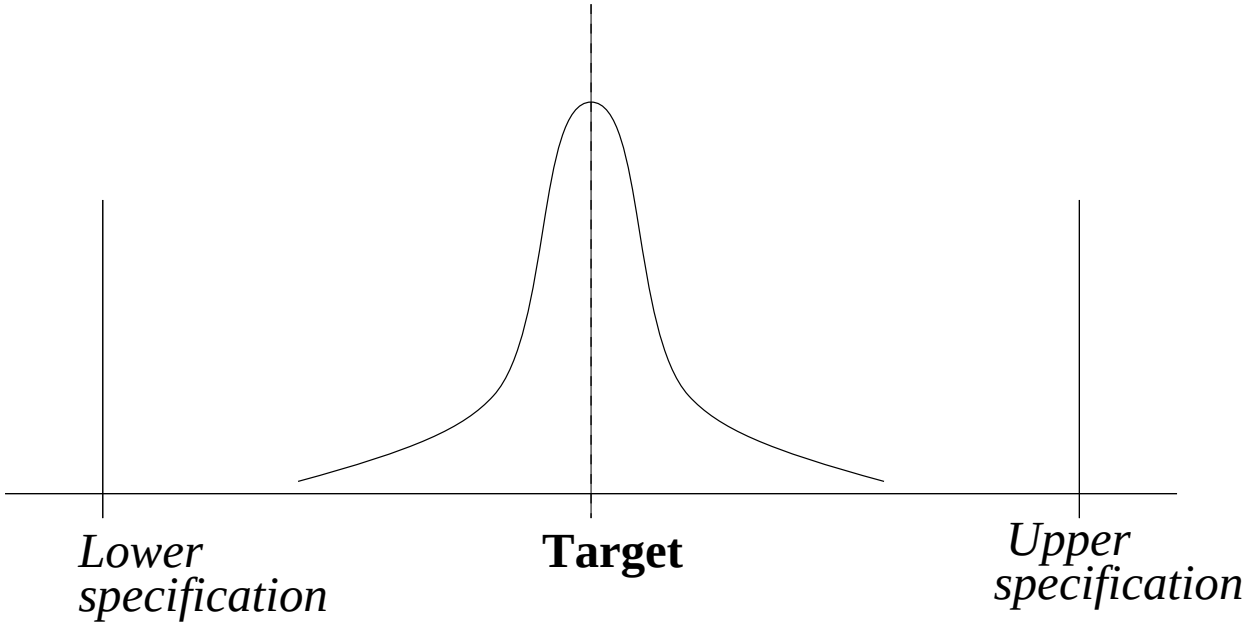
Produkce řízená inspekcí



Charakteristiky

- třídění/vyřazování produktů ležicích mimo povolený rozsah.
- $\pm$ specifikace

Produkce řízená cílem

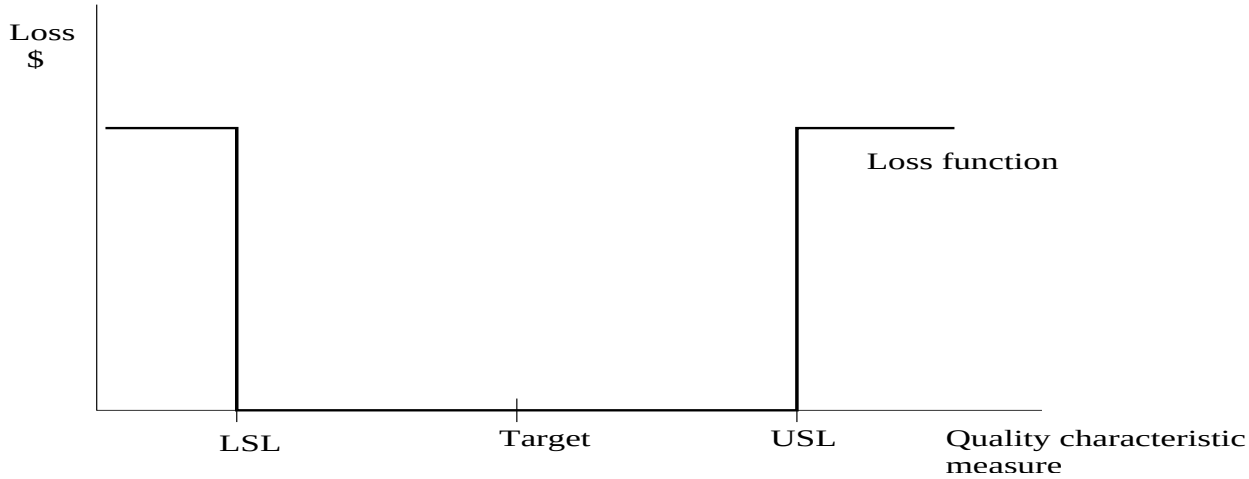


Charakteristiky

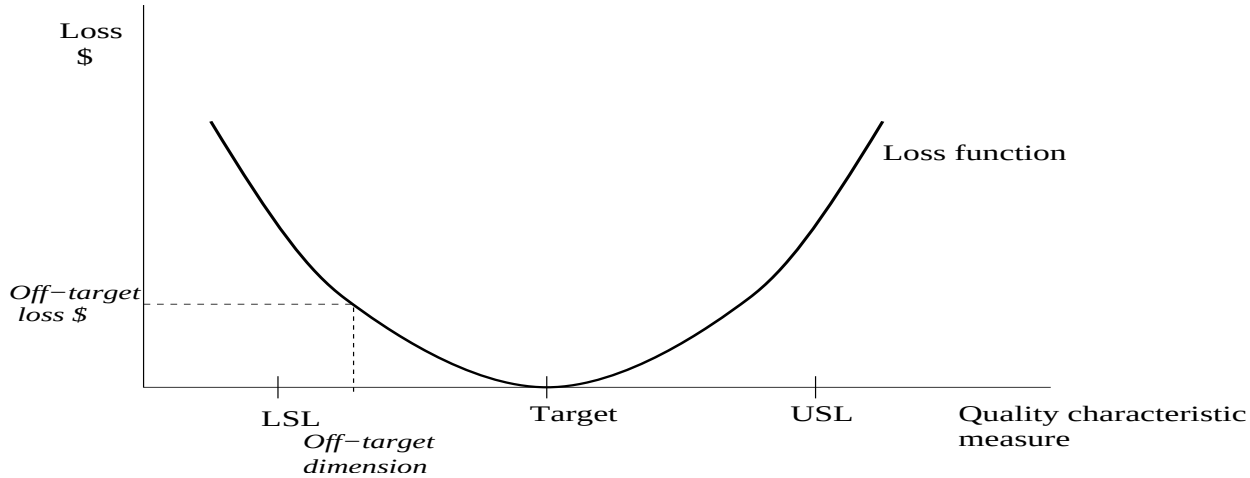
- zaměření na pozici cílového produktu a na redukci / řízení variace
- 6 sigma strategie uvedená Motorolou
  - specifikační omezení produktu je ve vzdálenosti $\pm$ 6 násobku standardní odchylky produkce
  - 2 defekty na miliardu produktů (za předpokladu normálního rozložení)
  - 3.4 či méně defektů na milión produktů při $\pm 1.5\sigma$ posunu středu

Ztrátová funkce kvality

Ztrátová funkce inspekční strategie



Ztrátová funkce cílové strategie



př. SONY v USA a Japonsku s rozdílnou kvalitou produktů

Kvadratická ztrátová funkce [Tag86]

- y ... produkovaná hodnota výkonnostního indexu,
- m ... hodnota indexu výkonosti požadovaná zákazníkem,
- $L(y)$ ... ztrátová funkce vzhledem k rozdílu mezi y a m,
- $L(y)$ může být rozložena do Taylorovy řady okolo m:

$$\begin{aligned} L(y) &= L(m+y-m) \\ &= L(m) + \frac{L'(m)}{1!}(y-m) + \frac{L''(m)}{2!}(y-m)^2 + \cdots \end{aligned}$$

- za předpokladu $L(m) = 0$,
- $L(y)$ je minimální při $y = m$, $L'(m) = 0$,
- ztráta může být aproximována:

$$L(y) \approx k(y-m)^2$$

- k je neznámý koeficient,
- K určení k je potřeba vědět ztrátu D způsobenou odchylkou $\Delta = y - m$.

$$k = D/\Delta$$

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.

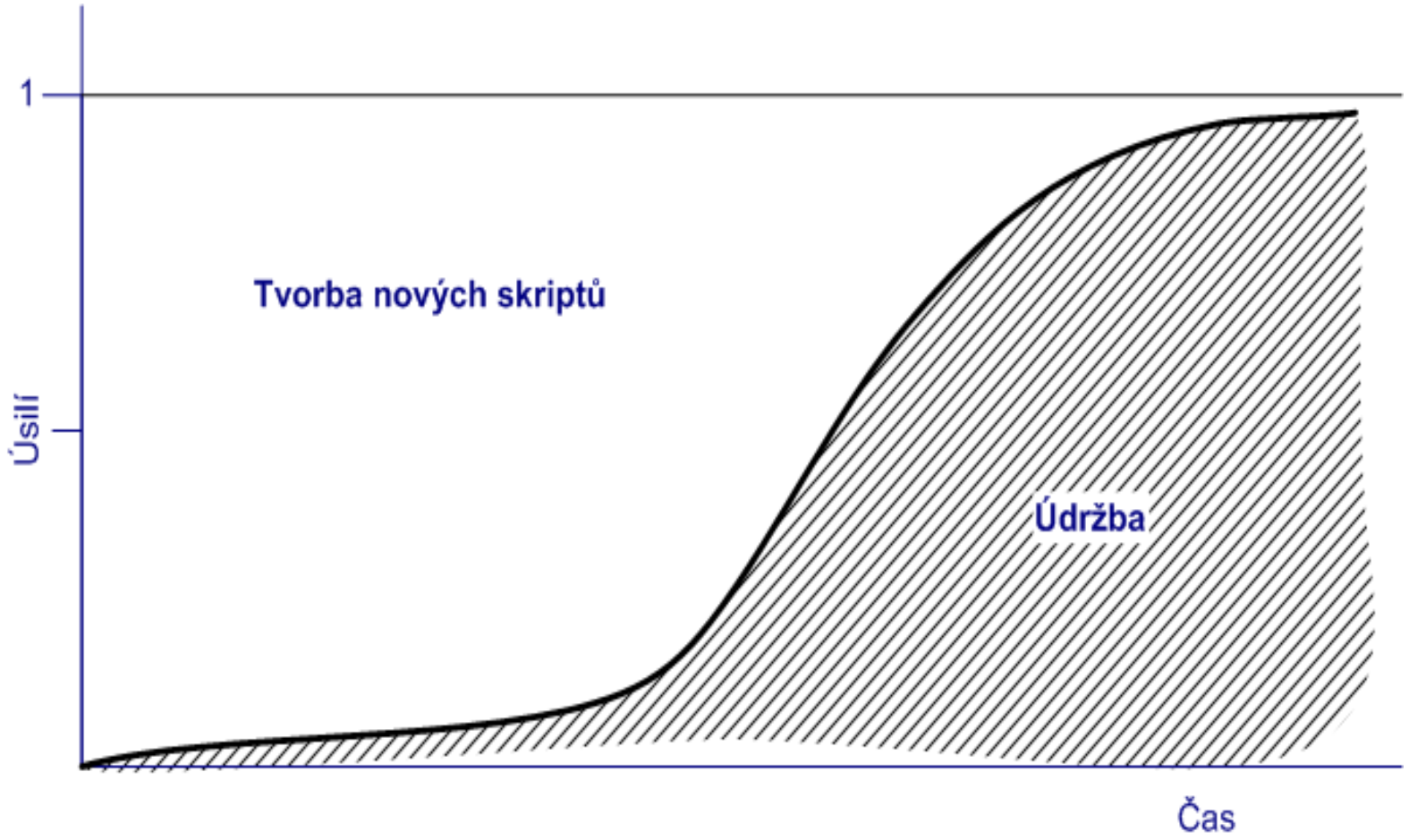
Problémy

1. Nereálná očekávání.
2. Slabá testovací praxe.
3. Očekávání, že automatizovaný test nalezne mnoho nových defektů.
   - typicky 85% manuální testování či návrh skriptu
4. Údržba automatizovaných testů.

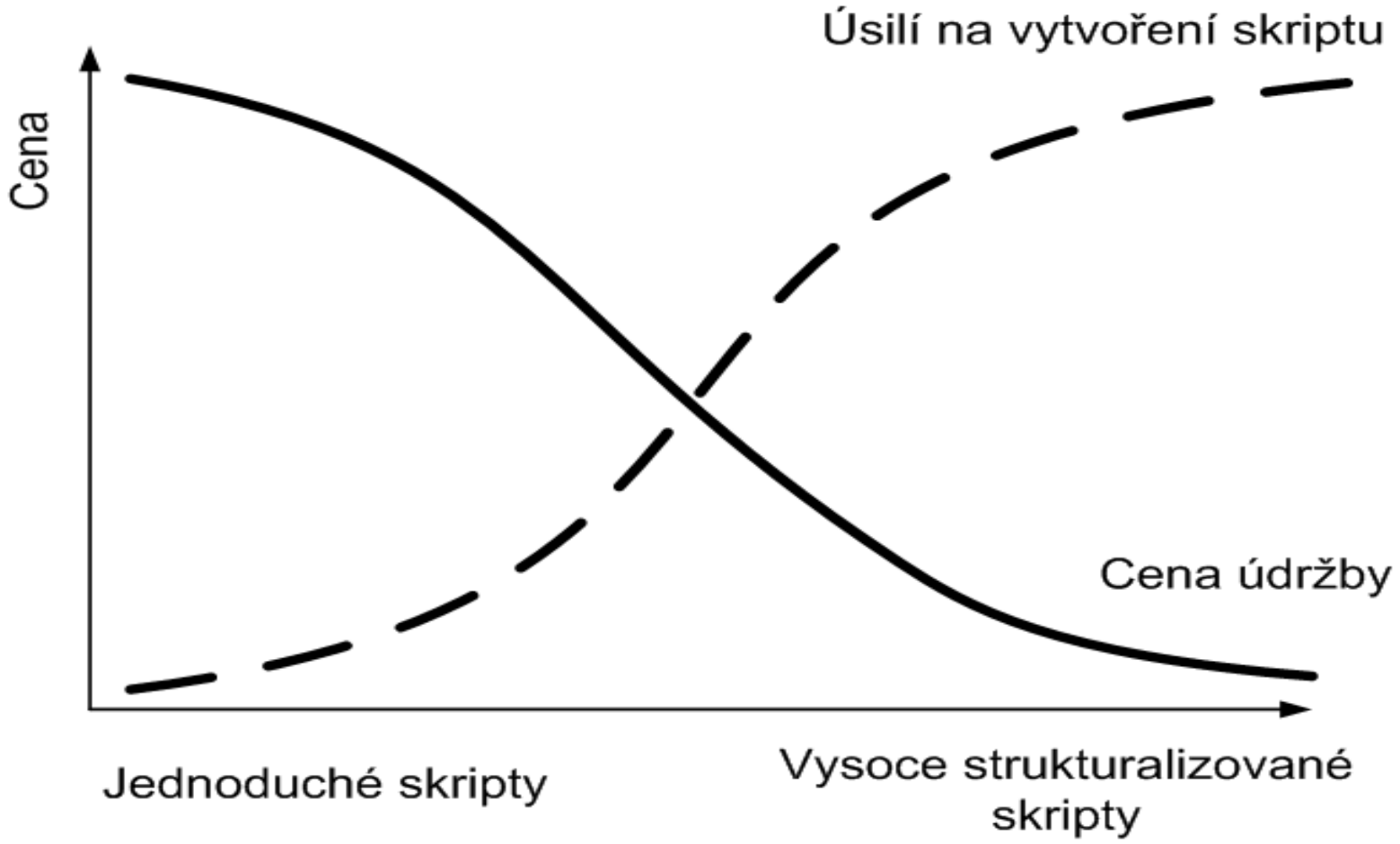
Porovnání postupů

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

Problém údržby



Úsilí vývoje



Literatura I

Cem Kaner, Jack Falk, and Hung Quoc Nguyen.
Testing Computer Software.
International Thomson Computer Press, second edition, 1993.

Edward Kit.
Software Testing in the Real World.
Addison-Wesley, 1995.

William J. Kolarik.
Creating Quality: Concepts, Systems, Strategies, and Tools.
McGRAW-HILL, INC., 1995.

Genichi Taguchi.
Introduction to Quality Engineering.
Asian Productivity Organization, 4-14, Akasaka 8-chome, Minato-ku, Tokyo 107, Japan, 1986.