

Semestrální práce ke kurzu 4IT421 Zlepšování procesů budování IS	
Semestr	ZS 2017/18
Autoři – jméno, příjmení , xname	Vojtěch Hyvnar, xhyvv00 Simona Madhi, xmadh00 Anna Pagáčová, xpaga00
Téma	Monte Carlo Planning Improves Decision Making
Datum odevzdání	17.12.2017

Abstrakt

Cílem této semestrální práce je představit simulační metodu Monte Carlo a její výhody při využití pro rozhodování v rámci procesů budování informačních systémů. Metoda Monte Carlo bude představena, popsána její historie a vlastnosti. Dále bude provedena ukázka a porovnání Weibullových distribučních funkcí, a předvedena simulace na praktickém příkladu. Součástí práce je také popis pozitivních a negativních aspektů simulace a samotné předvedení využití simulace při procesu plánování v agilní metodice Scrum.

Úvod

Dnešní procesy metodik agilního vývoje softwaru jsou velmi sofistikované, přesto ale dochází ke zjištění, že se často spoléhá na lidské subjektivní odhady, a to jak při výběru vhodné metodiky, tak i při výpočtu data dokončení projektu. Simulační metoda Monte Carlo nabízí možnou odpověď, jak vhodně dospět rozhodnutím pomocí kvantifikovatelného přístupu.

Cílem semestrální práce je seznámení čtenáře (posluchače) s tím, jak se využívá statistiky a metody Monte Carlo pro plánování a rozhodování během budování a vývoje informačních systémů a názorné předvedení aplikace metody na příkladu.

Cíl bude dosažen analýzou zdrojů a představení problematiky v sobě navazujících částech, ve které budou obsaženy praktické příklady a ilustrace.

Klíčová slova

Monte Carlo, metoda Monte Carlo, plánovací procesy metodiky Scrum, Weibullovo rozdělení

Obsah

Abstrakt	1
Úvod	1
Klíčová slova	1
1.1 Úvod do problematiky	3
1.2 Demonstrace nepřesnosti lidských odhadů	3
1.3 Úskalí klasického odhadu trvání projektu budování softwaru	4
1.4 Výhody využití metody Monte Carlo oproti klasickým plánovacím procesům Scrum	4
2 Metoda Monte Carlo	5
2.1 Historie	5
2.2 Podstata	5
2.3 Simulace	6
Časté chyby:	6
2.4 Výhody a nevýhody využití metody MC	6
3 Weibullovo rozdělení a praktická ukázka využití metody Monte Carlo	7
3.1 Weibullovo rozdělení	7
3.2 Výběr správného modelu	9
3.3 Faktory způsobující šikmost a asymetrii pravděpodobnostních rozdělení.	10
3.4 Příklad, jak dojdeme z grafického znázornění normálního rozdělení, k Weibullově funkci..	11
3.5 Podoby Weibullova rozdělení v závislosti na použitých metodikách	13
3.6 Predikce na základě historických dat	13
3.7 Jednoduchý příklad aplikace Monte Carlo ve Scrumu	15
Závěr	16
Citovaná literatura	17

1.1 Úvod do problematiky

IT vedení firem často zahajují zlepšování budování procesů pro vývoj softwaru na základě nespokojenosti s časovou a finanční náročností vývoje. Manažeři zodpovědní za směr produktu mívají spoustu nápadů, ale mívají pocit, že těmto nápadům trvá příliš dlouho, než se dostanou v konečné podobě zákazníkovi do rukou. Následně tito manažeři často získají frustrující pocit, že jejich konkurenti se buď zdají nebo skutečně jsou rychlejší v nabízení hotových produktů na trh než je jejich organizace, a hledají způsoby, jak zvýšit svou konkurenceschopnost. (Magennis, 2015)

Prosazovatelé a prodejci procesních změn nabízí škálu různých nejnovějších nástrojů, knih, školení a certifikací jako rychlou záplatu na neurčité, blíže nespecifikované problémy přílišné pomalosti či drahosti. XP, Agile, Lean, Scrum a Kanban jsou jedny z nejznámějších procesních metodik, které se dostaly na přední příčky popularity, všechny dodávané s případovými studiemi dokazujícími velký zlepšující vliv při jejich správné implementaci. Konečný výběr je však zdá se řízen subjektivní vírou či pocity, s mnoha organizacemi přeskakujícími od jedné metodiky k druhé ve snaze najít ideálního stavu. Pro informované rozhodování je zapotřebí kvantitativního rámce, který by odhadnul a ohodnotil skutečný dopad.

Měření kvantitativního dopadu procesu budování softwaru je obtížné. (Magennis, 2011) Měřitelné změny trvají týdny až měsíce, a použití kontrolní skupiny není dost dobře možné – změna je implementována a výsledky v případě neimplementace změny není zajímavá či snadno odhalitelná metrika. Tato práce nabízí jednu unikátní techniku kvantitativního odhadu potencionálních ekonomických dopadů jak před tak i po implementaci změny.

Základem zde popisované metody je probablistická simulace dopadu změn ve vzorcích „časových cyklů“ od předcházejícího projektu k projektu konečnému s využitím nové metodologie. Pro vytvoření odhadu potencionálního zisku nové metodiky je možné snížit existujících vzorky časových cyklů o fixní procentuální část, za účelem simulace finančního návratu při hypotetických sníženích (např. 10%, 25%) Jakmile změna nastane, lze porovnat skutečné výsledky s predikovanými daty k zhodnocení rozdílů a vylepšení modelovacích způsobů pro budoucí iniciativy.

V IT procesech neexistuje konsistentní definice začátku a konce pracovní činnosti. Definice se často různí mezi jednotlivými odděleními a týmy organizace. Časový cyklus je dobrým kandidátem metriky pro predikci a porovnávání, protože jde o uběhnutou časovou míru, která není měřená v abstraktních jednotkách (jako např. funkční body, příběhové body a rychlost, které jsou běžné u agilních metodik).

Magennis odlišuje dva pojmy časových cyklů – lead-time a cycle-time (Magennis, 2015). Lead-time je definován jako čas mezi vytvořením popisu práce a předáním práce do produkce. Cycle-time je pak definován jako čas mezi tím, kdy je činnost skutečně fyzicky zahájena na pracovní součásti, až po čas, kdy je považována za kompletní.

1.2 Demonstrace nepřesnosti lidských odhadů

De la Mazza navrhuje jednoduchý test přesnosti či nepřesnosti lidských odhadů (de la Mazza, 2017):

1. Vyberte položku k ohodnocení. Na týmové úrovni se bude nejspíše jednat o User Story, na programové úrovni o iniciativu nebo projekt.

2. Vyberte skupinu expertů, kteří provedou odhad. NA týmové úrovni půjde téměř určitě o lidi, kteří budou tuto práci vykonávat. Na programové úrovni půjde jespíše o skupinu senior engineerů.
3. Rozdělte tuto skupinu expertů na dvě části, a nechte je ohodnotit pracovní položku zvlášť.
4. Definujte měření chyby, například: Nechť je m průměr dvou odhadů, a x_2 ten vyšší ze dvou odhadů. Pak metrika chyby je $x_2/m - 1$.
5. Zeptejte se účastníků, jak špatná může metrika být, než cena tvorby odhadu překročí její hodnotu. Pro metriku zmíněnou výše to bývá většinou mezi 10 a 25 %.
6. Vypočítejte metriku a diskutujte výsledky.

1.3 Úskalí klasického odhadu trvání projektu budování softwaru

Standardní přístup v agilní metodice Scrum k plánování Product Backlogu a Sprintů se silně opírá o odhadování. (de la Mazza, 2017) Techniky přidělování Story bodů spočívají v odhadování náročnosti jednotlivých položek. Takový bod je bezrozměrná veličina, která se dále využívá při vyhodnocování a přidělování jednotlivých položek/úkolů týmům pro jednotlivé sprinty.

Jedním z problémů je, že tento přístup nelze vhodně aplikovat na vícero paralelně pracujících týmů, ani když mezi nimi nejsou žádné závislosti. Dále se také celou dobu počítá se zprůměrovanými hodnotami, bez ohledu na varianci.

1.4 Výhody využití metody Monte Carlo oproti klasickým plánovacím procesům Scrum

Pravděpodobnostní předvídání poskytuje přesné předpovědi ceny, rozpočtu a data dokončení s využitím méně odhadů. Historická data provedení jsou využita k vedení modelů, pokud je to tak možné. Přírůstek využití předpovědí Monte Carlo spočívají v tom, že úroveň přesnosti předpovědí napomáhá provádění informovaných rozhodnutí a postupování s jistotou.

Metoda Monte Carlo produkuje pravděpodobnosti: např. „Je 80% šance, že produkt bude dokončen mezi 18. a 25. lednem“ nebo „Je 20% šance, že rozpočet bude překročen o 20 % či více“. (de la Mazza, 2017)

2 Metoda Monte Carlo

2.1 Historie

Technika, v který se sleduje velké množství náhodně vygenerovaných čísel pomocí modelu pravděpodobnosti, aby se našlo přibližné řešení k číselnému problému, který by bol těžký vyřešit jinými metodami. (Oxford Dictionaries, 2017)

Základný princip metody Monte Carlo bol vyslovený už v 18. století francouzským přírodovědcem a matematikem Georges Louis Leclerc Comte de Buffonom . Úloha, dnes známa jako Buffonova jihla spočívá v tom, že pomocí náhodného házení jihly na papír s namalovanými rovnoběžkami ve vzdálenosti délky jihly, možno určit hodnotu čísla π . Sleduje se počet překřížení jihly s některou z rovnoběžek, potom $p = \frac{2}{\pi}$. (Vysoká škola báňská, 2017)

Metoda Monte Carlo bola formulovaná a súčasné využitá během druhý světový vojny vědeckými pracovními Stanislawom Ulamom a Johnom von Neumannom v Národní laboratoři Los Alamos v Spojených státech amerických při vývoji atomový bomby v rámci projektu Manhattan. Podnět k vytvoření metody pocházel z Ulamoveho zájmu o náhodné procesy, od nejjednoduchších po vznešené. Vytvořil pojem „šťastné čísla“, kterých distribuce se podobala prvočíselným číslům. Svoji zaujatostí k teorii větvení významně přispěl k jejímu rozvoji a uplatnění během vojny. (Metropolis, 1987)

Při výzkumu chování neutronu a možnostmi jeho pronikání různými látkami se vyskytl problém, jak určit procento neutronů v určité spršce, která pronikne například nádrží vody určitých rozměrů. Neexistovala metoda, která by předpověděla další chování jednotlivého neutronu. Vědci využili k modelování předpovědi "historie života neutronu" techniku kola rulety. A protože ruleta je typická pro Monte Carlo, Neuman pojmenoval využitou metodu velmi výstižným Název: METODA MONTE CARLO. (Vysoká škola báňská, 2017)

Pomocí této metody bylo možné předpovědět trajektorii každého neutronu daného svazku. Při předpokladu, že při srážce neutronu a atomu vodíku je neutron pohlcen průměrně v jednom ze sta případů. Při učení toho, zda bude neutron pohlcen nebo ne, je možno použít kolo rulety rozdělené na 100 dílků, přičemž 1 označený dílek bude znamenat pohlčení neutronu. V případě, že nedojde k zániku neutronu, se pomocí dalšího kola rulety náhodně stanoví trajektorie neutronu do další srážky. Simulace "historie života" neutronu se provádí tak dlouho, dokud není pohlcen, nebo dokud neztratí tolik energie, že jeho konečné odletět nás přestane zajímat, nebo dokud se mu bez újmy podaří vyjít z nádrže (Dorda, 2017)

2.2 Podstata

Metoda Monte Carlo je komplex algoritmů používaných k simulací možné skutečnosti. Je to stochastická simulační metoda, během simulace Monte Carlo se hodnoty náhodně odebírají ze vstupních pravděpodobnostních rozdělení. Každá sada vzorků se nazývá iterace a výsledný výsledek z tohoto vzorku se zaznamená. Simulace se zopakuje stovky nebo tisíce krát a výsledkem je rozdělení

pravděpodobnosti možných výsledků. Po získání obecních charakteristik bude možné podat popis namodelovaného jevu. Jednotlivé výsledky simulací nemají velkou vypovídací hodnotu, avšak několikanásobné opakování simulace přináší použitelné výsledky, díky platnosti zákona velkých čísel. Tímto způsobem simulace Monte Carlo poskytuje mnohem komplexnější pohled na to, co se může stát a s jakou pravděpodobností se to stane. Umožňuje získání alespoň přibližného výsledků v situacích, kdy jsou běžné analytické přístupy příliš složité či nemožné. (Simulace.info, 2015)

2.3 Simulace

Obecný postup při řešení úloh metodou Monte Carlo: (Palisade, 2017)

1. *vymezení cíle*
2. *příprava dat a výběr nástroje*
3. *analýza problému*
4. *stanovení pravděpodobnostního rozdělení náhodných veličin*
5. *vytvoření modelu*
6. *vygenerovaný mnoha scénářů (simulace)*
7. *statistické zpracování výsledků*

Na začátek si určíme cíl, proč a za jakým účelem simulaci vytváříme a co ní chceme dosáhnout. Budeme potřebovat kvalitní soubor dat, ze kterých se bude v simulaci vycházet. Dalším krokem je vybrat vhodný nástroj pro simulaci, např. MATLAB, SAS z volně dostupných MS Excel.

Při rozboru problému a návrhu modelu je důležitým krokem zvolit klíčové parametry, určit náhodné veličiny a vybrat vhodné pravděpodobnostní rozdělení vycházející z historických dat.

Po vytvoření modelu následuje generování velkého množství scénářů. Doporučuje se nejprve vygenerovat menší počet scénářů abychom se ujistili, zda simulace pracuje správně. Generováním náhodných kombinací hodnot vstupních náhodných veličin podle jejich pravděpodobnostního rozdělení získáme scénář. Každý z scénářů představuje jeden možný vývoj reálné situace. Na závěr statisticky zpracujeme výsledky a vyvodíme závěry.

Časté chyby:

- nevhodný výběr pravděpodobnostního rozdělení
- špatně navržený generátor náhodných čísel
- vyvození závěru z malého počtu opakování
- nezahrnutí důležité veličiny do simulace

2.4 Výhody a nevýhody využití metody MC

Výhody: (Palisade, 2017)

- **pravděpodobnostní výsledky**
Výsledky ukazují nejen to, co se může stát, ale také jak moc je pravděpodobné, že každý výsledek nastane.
- **grafické výsledky**
Díky údajům, které generuje simulace Monte Carlo, je snadné vytvořit grafické pomůcky, různých výstupů a jejich šance výskytu, které jsou užitečné hlavně při komunikaci s nezasvěcenými stranami.
- **analýza citlivosti**
Deterministická analýza jen v málo případech ztěžka vidí, které proměnné nejvíce ovlivnily výsledek. V simulaci Monte Carlo je snadné zjistit, které vstupy měly největší vliv na výsledky.
- **analýza scénáře**
V deterministických modelech je velmi těžké namodelovat různé kombinace hodnot pro různé vstupy, abychom viděli skutečné účinky odlišných scénářů. Pomocí simulace Monte Carlo mohou analytici přesně zjistit, které vstupy měly hodnoty, které se společně vyskytly při výskytu určitých výsledků. To je neocenitelné pro další analýzu.
- **korelace vstupů**
V simulaci Monte Carlo je možné modelovat vzájemně závislé vztahy mezi vstupními proměnnými

Nevýhody: (Vysoká škola báňská, 2017) (Palisade, 2017)

- **Relativně malá přesnost**
Jelikož se přesnost metody odvíjí i od počtu simulací, doporučuje se započítávat chyba = $\sqrt{\frac{B}{N}}$, kde N počet náhodných experimentů (simulací) a B je konstanta vyjadřující povahu konkrétního příkladu. Pro zvýšení přesnosti výsledku o jednu řadu je třeba zvýšit počet simulací o 2 řady
Přesnost odhadu nezávisí jen na celkovém počtu simulací N, ale také na řadě určované pravděpodobnostní poruchy Pf. Definice variačního koeficientu pravděpodobnosti poruchy pro malé pravděpodobnosti: $v_{pf} = \frac{1}{\sqrt{N * P_f}}$
- **silná závislost na historických datech**
Kvalita simulace je závislá na přesnosti a správnosti dat. Pokud je systém například nestabilní, odebrání vzorků z historických údajů přináší špatné výsledky.
- **vytváření modelu**
obtížnost stanovení pravděpodobnostního rozdělení náhodných faktorů spolu s respektováním jejich závislostí

3 Weibullovo rozdělení a praktická ukázka využití metody Monte Carlo

3.1 Weibullovo rozdělení

Dříve než začneme vytvářet jakékoli predikce výše zmíněnou metodou (a platí to samozřejmě i u jiných předpovědí), je nutno určit, podle jakého modelu je nejvhodnější tyto předpovědi tvořit. Vhodný model se vybírá často na základě analýzy historických dat. Nejprve zde představím k predikcím často používané Weibullovo rozdělení a následně se pokusím vysvětlit, proč právě toto pravděpodobnostní rozdělení je nejvhodnějším modelem pro námi tvořené predikce.

Tento teoretický model je pojmenován podle švédského profesora Waloddi Weibulla na základě jeho článku *A Statistical Distribution Function of Wide Applicability* z roku 1951. V tomto článku Weibull na několika rozdílných příkladech ukázal možnost širokého využití svého modelu. Mezi tyto příklady patřilo například testování odolnosti oceli nebo síly vláken Indické bavlny. Dá se říci, že se nejčastěji jedná o metodu kvantitativní analýzy spolehlivosti testovaných položek, která je založena na aplikaci metody statistického modelování.

V době, kdy Weibull tento článek psal, byla tvorba softwaru tak, jak ji známe dnes, ještě v plenkách, ale jak se později ukázalo, možnosti tohoto modelu jsou opravdu široké, a proto se Weibullovo rozdělení dnes používá i při aproximaci odhadů délky trvání procesů tvorby softwaru nebo také při testování bezporuchovosti elektronických systémů či komponent. Obecně toto rozdělení modeluje datové soubory, jejichž hodnoty jsou větší než nula.

V základní podobě má funkce hustoty pravděpodobnosti Weibullova rozdělení 3 parametry a její předpis vypadá následovně:

$$f(x) = \frac{\beta}{\eta} \left(\frac{x - \gamma}{\eta}\right)^{\beta-1} * e^{-\left(\frac{x-\gamma}{\eta}\right)^\beta}$$

- β**
- Parametr sklonu nebo tvaru (anglicky shape)
 - Tento parametr nám udává tvar nebo průběh dané funkce hustoty pravděpodobnosti.

Tento parametr v klasickém využití nejčastěji symbolizuje četnost selhání (anglicky failure rate) či poruchovost. Obvykle nabývá hodnot mezi 0,5 a 10.

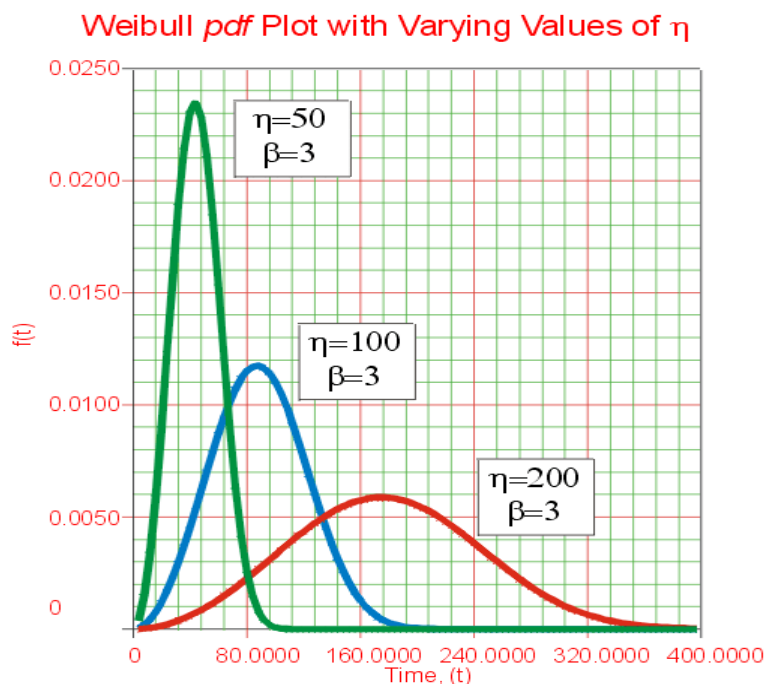
- Pokud je roven 1, znamená to, že se četnost selhání s časem nemění.
- Pokud je menší než 1, značí to, že četnost selhání s časem klesá.
- Naopak pokud je větší než 1, četnost selhání s časem roste.

- η**
- Parametr měřítka (anglicky scale)
 - Tento parametr mění měřítko na časové ose, například počet hodin či velikost délky cyklů
 - Změna tohoto parametru nemá žádný vliv na tvar rozdělení, pouze mění velikost hodnot na ose

- Velikost tohoto parametru se dá aproximovat z empirických hodnot jako 63,2. percentil všech hodnot. Pravdepodobnosti v tomto bodě nezávisí na parametru sklonu

- γ**
- Parametr umístění nebo polohy (anglicky location)
 - Tento parametr udává minimální hodnotu náhodné veličiny.
 - Parametr umístění se v praxi často nepoužívá a proto se nahrazuje nulou a funkce hustoty pravdepobonosti Weibullova rozdělení má pak efektivně pouze dva parametry.

Pro správné použití Weibullový funkce je potřeba vypočítat potřebné parametry. Toto se provádí buď pomocí teoretických metod, například metodou maximální pravděpodobnosti nebo metodou nejmenších čtverců za použití Weibullova pravděpodobnostního grafu a nebo za pomoci různých statistických softwarů.

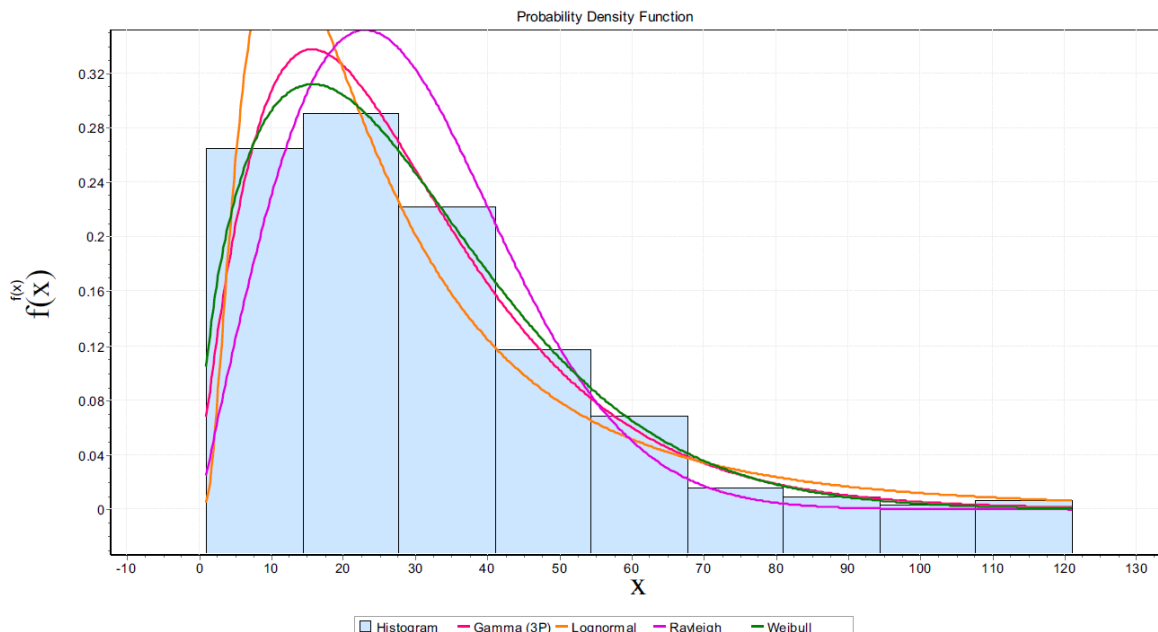


Příklad tvaru grafu Weibullový funkce v závislosti na změně parametru měřítka při neměnném parametru sklonu.

3.2 Výběr správného modelu

Už empirické průzkumy 60. a 70. let 20. století ukázaly, že se tvorba softwarových projektů, narozdíl například od klasických manufakturních procesů (u kterých lze díky centrální limitní větě většinou pozorovat Gaussovo normální rozdělení), řídí Rayleighovým rozdělením pravděpodobnosti, což se dá považovat za zvláštní případ Weibullova rozdělení, kdy je parametr sklonu roven dvěma.

Analýza histogramů rozdělení četností cycle-times při vývoji komerčních softwarů ukázala převahu levostranných, kladně sešikmených grafů. Pokud v grafu cycle-times vývoje softwaru tento vzor šikmosti chybí, může to indikovat špatně zachycená empirická data. Histogramy takového tvaru mohou být funkcí několika pravděpodobnostních rozdělení. Může se jednat například o Gamma, Weibulovo, Rayleighovo či Logaritmicko – normální rozdělení.



Srovnání několika pravděpodobnostních rozdělení a jejich křivek hustoty pravděpodobnosti v porovnání s histogramem empiricky naměřených časů trvání jednotlivých cyklů při tvorbě softwaru.

Histogram empirických hodnot je proložen několika pravděpodobnostními funkcemi, ze kterých je potřeba vybrat tu, která nejlépe odpovídá reálným hodnotám. Pomocí vybrané funkce poté můžeme modelovat a předpovídat délku cycle-times při vývoji softwaru. Pro předpovědi budoucích časů či dat je nejbezpečnější zvolit tu funkci, která je „nejdelší“. Z výše zmíněných funkcí má nejdelší chvost Weibullova funkce, velikosti četností s časem klesají pomaleji, než u ostatních funkcí. To znamená možnost předpovídat více zpoždění při vývojářských projektech.

Podle studií **McCombse** a kolektivu by vhodná funkce pro modelování trvání délky různých projektů (nejen při vývoji softwaru) měla splňovat několik předpokladů:

- být spojitá
- mít počáteční a koncový bod
- mít definovaný modulus mezi koncovými body
- mít schopnost popsat jak šikmé, tak symetrické distribuce času.

Weibullovo rozdělení splňuje všechny podmínky kromě druhé, jelikož nemá koncový bod. Tahle podmínka však nemusí být vždy nutná. Při vývoji softwaru se mohou objevovat dlouhé časové prodlevy, které však mívají minimální pravděpodobnost, a proto se jejich vliv dá považovat za zanedbatelný. Zároveň se dá předpokládat, že pokud se objeví nezvykle dlouhotrvající proces, je na něj zaměřena pozornost a je vyřešen, případně ukončen.

3.3 Faktory způsobující šikmost a asymetrii pravděpodobnostních rozdělení.

Při procesech vytváření softwaru se berou velké, obecné nápady a požadavky, které jsou rozdělovány do menších dílů, a ty pak postupně implementovány. Tyto menší díly jsou v závislosti na použité metodě (Scrum, Kanban, XP a další) různě pojmenovány. Souhrnně lze označit jako pracovní položky (anglicky Work items). Tvorba každé pracovní položky prochází několika kroky a výsledek je narozdíl třeba od manufakturní tvorby unikátní.

Proces vývoje softwaru tedy nelze modelovat jako manufakturní proces. Klasický manufakturní cycle-time procesu se na základě centrální limitní věty bude blížit Normálnímu rozdělení. Toto je

způsobeno podrobným zkoumáním výsledků tvorby, kdy vše musí podléhat standardizaci. Délky cycle-times musí spadat do určitého intervalu ohraničeného stanovenými směrodatnými odchylkami, které je snaha minimalizovat.

Snaha o standardizaci v odvětvích, která k tomu nejsou stavěné, potlačuje kreativitu při tvorbě a snižuje kvalitu výsledných produktů. Délka práce na jednotlivých pracovních položkách, které jsou založené na kreativitě, a neměly by tedy podléhat standardizaci (jako jsou v našem případě položky při vývoji softwaru), by měla být tedy ze zásady variabilní a nepodléhat Normálnímu rozdělení.

Příklady vlastností procesu vývoje softwaru, které jej odlišují od manufakturních procesů:

- Tvorba jednotlivých pracovních položek má různé podoby, nejedná se o jednu fixní cestu, jak výsledný produkt vytvořit
- Každá položka se může lišit v náročnosti na čas, znalosti či další vstupy
- Prodlevy, které se mohou při vývoji objevit, jsou různé a často ani nemusí být přímo spojeny s položkou, která je tvořena
- Dokončování položek může znamenat tvorbu nové práce – přehlédnuté požadavky, oprava možných chyb a jiné.

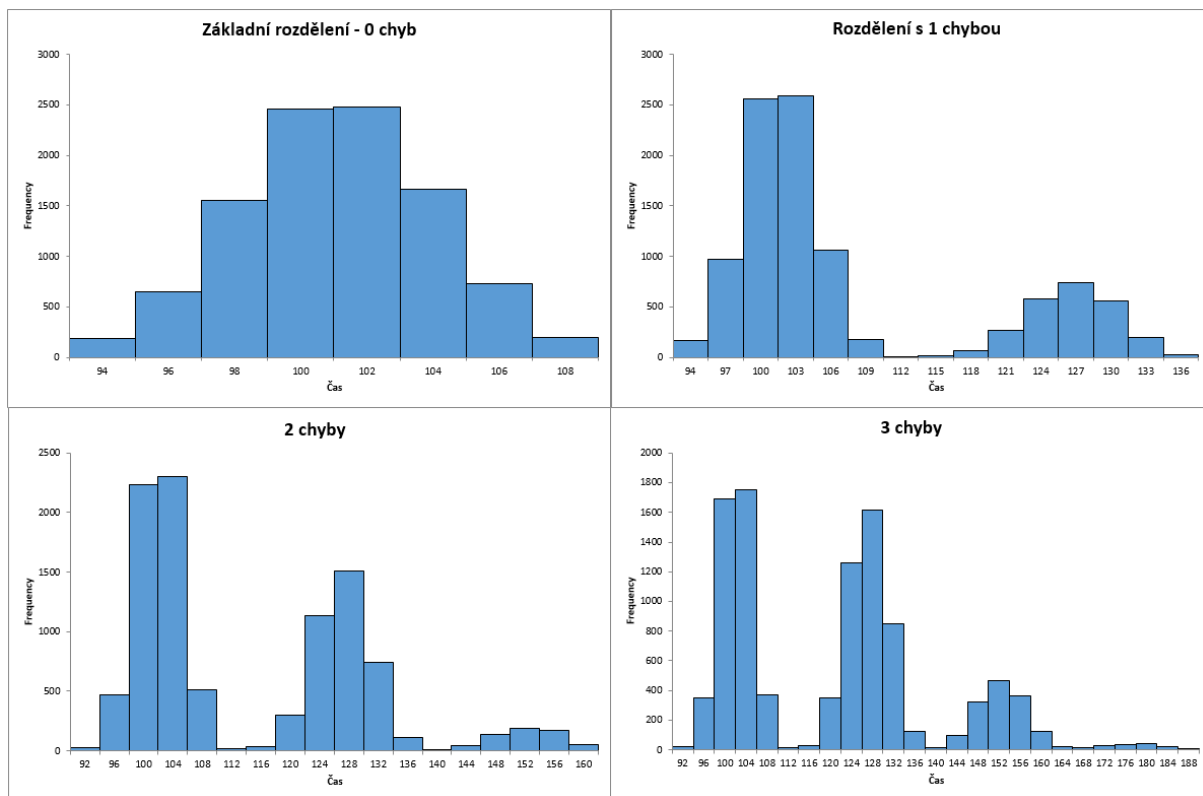
3.4 Příklad, jak dojdeme z grafického znázornění normálního rozdělení, k Weibullově funkci.

Jak bylo popsáno výše, Weibullova distribuce modeluje v podstatě bezporuchovost. Ať se jedná o odolnost oceli, sílu bavlny a nebo v našem případě šanci, že vše půjde podle plánu a produkt nebo pracovní položka bude vytvořen včas.

Jako základ použijeme funkci z normálního rozdělení se zvolenou střední hodnotou a směrodatnou odchylkou. V našem příkladě to je $N(100,9)$. Normální rozdělení je symetrické, četnosti generovaných hodnot jsou nejvyšší okolo střední hodnoty. Podobné rozdělení může mít například délka klasického manufakturního procesu. Všechny produkty jsou stejné, jejich tvorba je stejná a tudíž i délka vytváření je stejná. Jsou zde malé odchylky, které se management snaží eliminovat. Není zde žádný prostor pro kreativitu, všechno je předem dané, chyby jsou zde minimální. Stejná pravidla samozřejmě při tvorbě softwaru neplatí.

V našem příkladě budeme uvažovat možnost, že se k tomuto procesu, který má normální rozdělení, může přidat chyba (neboli zpoždění), která s určitou pravděpodobností zpozdí proces tvorby softwaru o určitou dobu. Zde použiju jednoduše 25% šanci, že se objeví chyba a pokud se chyba objeví, zpozdí tvorbu o 25%. Takto jednoduše to samozřejmě v praxi nefunguje, ale pro potřeby tohoto příkladu to stačí. Dále předpokládáme, že se chyba může objevit v jednom procesu více. V tomhle zjednodušeném příkladě by se tedy dvě chyby zároveň objevily s šancí 6,25%. Kombinace těchto chyb by zpozdila proces tvorby o 56,25%. Tyto chyby se nám mohou hromadit. V praxi jich může být mnoho a jejich závažnost být odlišná. V našem příkladě jsou, jak zmíněno výše, všechny stejně velké se stejnou pravděpodobností. Jakmile začneme přidávat více chyb, dostaneme se postupně do bodu, kdy procentuální zastoupení chyb je vyšší, než původní rozdělení bez chyb. Situace, kdy by se objevilo mnoho chyb najednou mají samozřejmě menší pravděpodobnost než původní jedna chyba. Tato nízká pravděpodobnost je v grafu zastoupena už výše zmíněným dlouhým chvostem.

V následném grafu lze vidět výše popsaná procedura s žádnou až třemi chybami. Prázdné intervaly, ve kterých jsou nulové nebo minimální četnosti, jsou způsobené tím, že závažnost chyby jsme zvolili ve všech případech stejnou. Při lišících se velikostech jednotlivých chyb by se prázdná místa zaplnila.



Tento model i když hodně jednoduchý, pomůže objasnit a vizualizovat podstatu problému – ukazuje, jak se mění délky cycle-times v závislosti na nejistotách a prodlevách, které se nemusí týkat samotných vytvářených položek. Může se čekat na dostupnost odborníka na danou problematiku, na to, než se připraví prostředí pro testování nebo na to, než se dokončí související položky. Tyto prodlevy tedy nemusí souviset se samotnou tvořenou položkou, může se jednat pouze o smůlu. Kombinace jenom pár těchto chyb či prodlev vytváří asymetrii v grafech. Je také velmi málo pravděpodobné, že by jednu položku ovlivnily všechny možné prodlevy.

Následující obrázek ukazuje velikost parametrů změřených z výše vytvořeného ukázkového modelu. V levé části je odhad parametrů z modelu v přítomnosti dvou a tří chyb za pomoci jednoduché funkce v jazyku R. V pravé části je pak ukázka aproximace parametru měřítka, jak bylo zmíněno v první kapitole, jako 63,2. percentil všech hodnot. Lze vidět, že jsou odhady parametru měřítka vcelku podobné.

Parametr sklonu, jak bude patrné z následující kapitoly, by měl v procesech tvorby softwaru nabývat ideálně hodnot mezi 1 a 2, což v našem příkladě neplatí. Toto je způsobeno jednoduchostí této ukázky. Důležité ale je, že s přidáním chyby parametr sklonu klesá a blíží se těmto nižším hodnotám.

```
> fit2chyby
  shape    scale
7.11132 119.76709
> fit3chyby
  shape    scale
6.255815 127.291007
> |
```

	63,2. percentile
2ch	121,919982
3ch	125,106222

Úplně přesná pravděpodobnost jednotlivých ovlivňujících jevů se nedá předpovědět, ale dá se měřit z historických dat. Analýza těchto pravděpodobností a jejich dopadu je velmi důležitá pro rozhodování v procesech tvorby a rozpočtování k řešení nejčastějších komplikací.

3.5 Podoby Weibullova rozdělení v závislosti na použitých metodikách

To, že je Weibullovo rozdělení nejspíše tím správným rozdělením cycle-times při softwarových procesech, podporuje fakt, že se tvar Weibullova rozdělení drží ve vývojářských procesech už od 70. let 20. století.

V 70. – 90. letech byl velmi rozšířený způsob vývoje pomocí vodopádového modelu. Dokud nebyla dokončena jedna etapa vývoje, nezačala druhá. Na základě empirických studií Norden, Putnam a Kan zjistili, že tento styl vývoje má Rayleighovo rozdělení. Toto rozdělení se dá brát jako speciální případ Weibullova rozdělení, kdy je parametr sklonu roven dvěma.

Agilní metodiky (XP, Scrum, Kanban), které se začaly rozšiřovat zejména v 90. letech minulého století a setrvávají dodnes, ukázaly posun Weibullový distribuce doleva, doprovázeno zúžením tvaru. To je způsobeno snížením parametru sklonu, který byl u těchto metodik naměřen v rozmezí mezi 1,3 a 1,6.

Nejnižší zjištěné parametry Weibullova rozdělení se blížily k jedné. Toto znamená přilížení exponenciálnímu rozdělení. Takové chování bylo napozorováno u týmů, které nejsou závislé na externích vlivech a jejichž práce je repetitivní. Tyto týmy nezažívají prodlevy díky externím vlivům, jejich práce není náročná na invenci, používají Lean praktiky - soustředí se na snížení work-in-progress a eliminaci nadbytečné práce.

Velikost parametru sklonu má velký vliv na rozptyl při předpovědích založených na analýze časových cyklů. Nižší parametr sklonu značí menší předpovídaný rozptyl a tudíž přesnější předpovídatelnost budoucích odhadů. Některé agilní praktiky tento parametr snižují pomocí častějších iterací, které snižují délku jednotlivých cyklů, jiné se zase snaží eliminovat nejistotu.

Následující tabulka shrnuje výše popsané vlastnosti týkající se velikostí parametrů.

Weibull Shape Parameter	1.3 to 2 (Weibull Range)	Traits: Small unique work items. Medium WIP. Few external impediments. Fair predictability.	Traits: Larger unique work items. High WIP. Low predictability. Many external dependencies. Process advice: Focus on identification and removal of impediments and delays, and quality. Scrum optimal.
	1 to 1.3 (Exponential Range)	Traits: Small or repetitive work items. Low WIP. Few external dependencies. Good predictability. Process advice: Automation of tasks, focus on task efficiency. Lean/Kanban optimal.	Traits: Larger work items. Large WIP. Many external dependencies. Poor predictability.
		0 to 10	10 to 30
Weibull Scale Parameter			

Parametr , který je nižší než 1, značí Hazardní funkci. Znamená to, že čím déle zůstává pracovní položka nedokončená nebo blokována, tím je vyšší šance na její dokončení. Dle empirických poznatků autorů studie je toto chování v praxi při vývoji softwaru velmi vzácné.

3.6 Predikce na základě historických dat

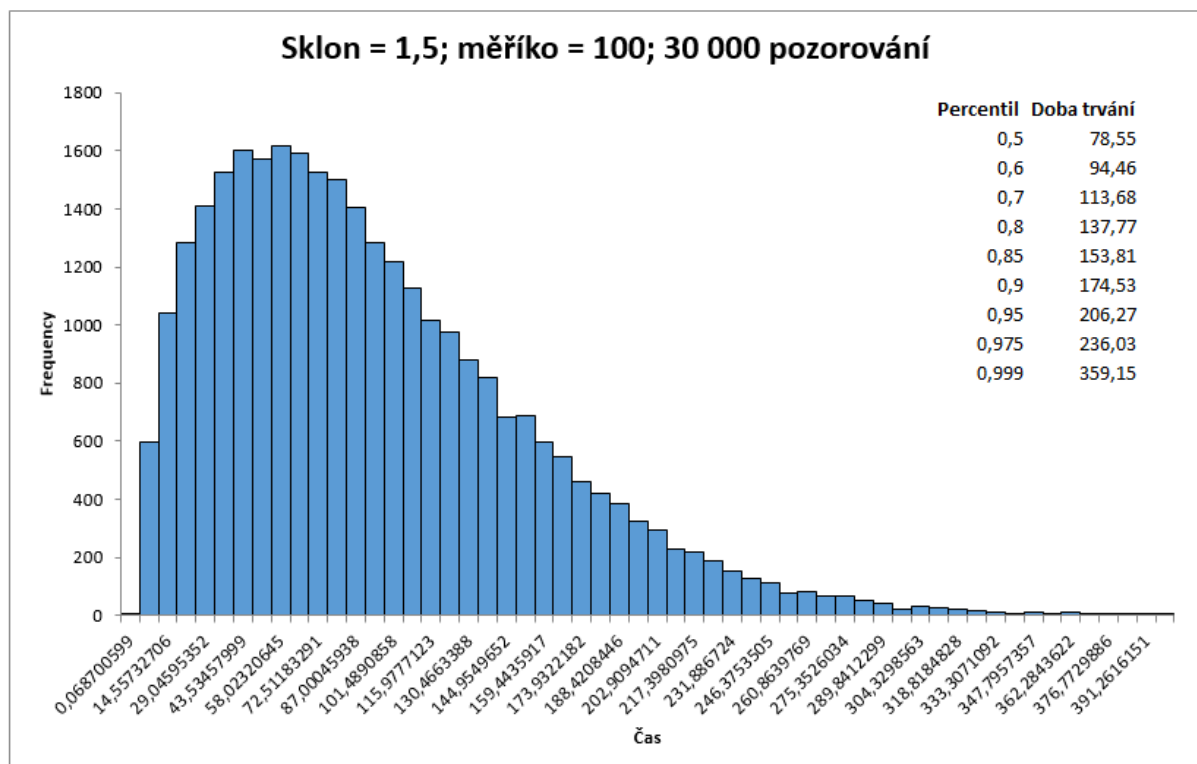
Pokud chceme předpovídat a modelovat budoucí vývoj a dopad délky trvání při tvorbě softwaru na základě předešlých, musíme mít k dispozici následující historická data:

- datum zahájení projektu
- datum ukončení projektu
- počet pracovních položek, které bylo potřeba dodat
- délka cycle-time pro každou položku

Když známe datum začátku nového projektu, který modelujeme, cílem je předpovědět datum ukončení projektu na zvolené hladině spolehlivosti. Často se používá hladina spolehlivosti 0,85, což znamená, že námi předpovězené datum ukončení projektu bude s 85% pravděpodobností podle modelu dříve než vypočtená hranice. Samotná předpověď pak bere náhodně vybrané sady historických délek trvání tvorby jednotlivých položek. Takových sad můžeme vybrat, pro co největší přesnost, tisíce. Na základě vybraných výsledků pak můžeme určit pravděpodobnost jednoho výsledku jako procento všech pokusů, které byly menší nebo stejné jako požadovaná hodnota.

Díky nedostupnosti dat, která firmy bohužel nezveřejňují, a která by byla potřebná pro reálnou simulaci v rámci této práce, použijeme pro predikci fiktivní hodnoty:

Z historických dat firmy bylo změřeno, že délka tvorby projektu (ve dnech) určitého typu má Weibullův parametr sklonu roven 1,5 a parametr měřítka roven 100 (parametr umístění neuvažujeme). Na základě těchto parametrů lze pomocí generování náhodných časů nového projektu určit pravděpodobnosti, s jakými bude projekt dokončen do určitého data (v našem případě – jak dlouho bude projekt trvat).



Na grafu jde vidět histogram četností jednotlivých předpovědí a také hladiny spolehlivosti. Pokud bychom vzali výše zmiňovanou hladinu spolehlivosti 0,85, můžeme říci, že s 85% pravděpodobností bude projekt dokončen za 154 dní. S 95% pravděpodobností pak bude projekt dokončen za 207 dní. Toto je velmi jednoduchý příklad, který je omezen zejména tím, že nemáme k dispozici historická data

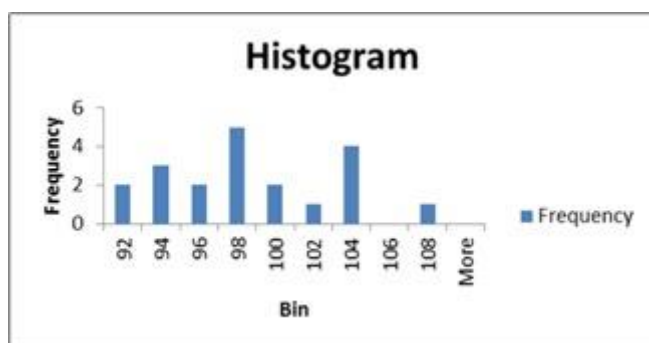
a také proporce budoucího projektu. Pokud bychom znali cenu projektu, případně výši fiktivních penalizací za nedodržení termínů, mohli bychom vyčíslit finanční dopady jednotlivých výsledků.

3.7 Jednoduchý příklad aplikace Monte Carlo ve Scrumu

Následující příklad byl publikován v článku Michaela de la Mazza. (de la Mazza, 2017)

Tento příklad už neuvažuje Weibullovo rozdělení, jelikož se jedná o rovnoměrnou distribuci.

Uvažujme Scrum tým, jehož velocity (schopnost dokončení bodů za jeden sprint) dosahuje hodnot mezi 80 a 120. Tým by chtěl vědět, kolik bodů bude dokončeno v šestisprintovém vypuštění do produkce. Pokud by tým pouze vypočítal průměrnou velocity, dospěl by k výsledku, že za šest sprintů dokončí 600 bodů. Naproti tomu, v grafu níže jsou výsledky simulace Monte Carlo s 20 vzorovými průběhy:



Tyto výsledky ukazují, že za šest sprintů je 10% pravděpodobnost, že velocity bude průměrně 92, o celých 8 % více, než se vypočítalo při využití analýzy průměrováním. Tato data navíc vedou k užitečným konverzacím s Product Ownerem: „Co se stane, pokud nižších 8 % backlogu nebude dokončeno do release date? Pokud je toto akceptovatelné, může být produkt vypuštěn dříve? Pokud to není akceptovatelné, jaké máme možnosti?“

Jak se tato pravděpodobnost vypočítala? Monte Carlo vzorkuje historickou distribuci, v tomto případě rovnoměrnou distribuci mezi 80 a 120. Pro získání grafu bylo vzorkování distribuce provedeno 20krát. Jeden ze vzorků byl následující:

100	120	118	112	91	110
-----	-----	-----	-----	----	-----

V tomto konkrétním vzorku velocity dosahovala hodnot od 91 do 120. Průměr byl 108,5. V grafu nahoře je tento vzorek zodpovědný za sloupec s hodnotou 108 na ose X.

Tady je další vzorek:

87	93	91	100	87	113
----	----	----	-----	----	-----

V tomto případě velocity dosahovala hodnot od 87 do 113 s průměrnou hodnotou 95.

Na každý vzorek je možné nahlížet jako na „možnou budoucnost“. Monte Carlo vytváří možné budoucnosti z historických informací a následně je zvažuje a posuzuje.

Jedna z největších výhod Monte Carlo je v tom, že funguje s jakoukoliv distribucí, i s těmi, které nejsou známy. V tomto jednoduchém příkladu byla distribuce známá a problém dostatečně

jednoduchý na to, aby pravděpodobnostní distribuce na konci šesti sprintů měla jasné, uzavřené řešení. Toto ale většinou neplatí.

Závěr

V úvodu článku představujeme situaci v odvětví a nastiňujeme problémy klasických přístupů k provádění rozhodování a přínosy využití simulace Monte Carlo.

V další kapitole jsme představili metodu Monte Carlo a její historii. Popsali jsme podstatu této metody, základní způsoby využití, jednotlivé výhody a častá úskalí.

Ve třetí části jsou popsány předpoklady pro zvolení správného modelu pro predikce budoucích hodnot na základě historických dat. Dále je zde popsána základní charakteristika Weibullova rozdělení, které bylo zvoleno jako nejlépe vyhovující. K tomu je přidána ukázka, jakým způsobem se tvar takového rozdělení vytvoří. Popisujeme také, jak vypadají a čím jsou tvořeny hodnoty jednotlivých parametrů tohoto rozdělení.

Na závěr jsme uvedli dva praktické příklady využití predikce za použití metody Monte Carlo. První příklad bylo obecné využití Weibullova rozdělení na základě historických dat. Druhý příklad se týkal jednoduché simulace sprintů ve Scrumu při znalosti průměrné rychlosti minulých sprintů.

Citovaná literatura

de la Mazza, Michael. 2017. Monte Carlo Planning Improves Decision Making. *Infoq*. [Online] 2017. [Citace: 24. 10 2017.] <https://www.infoq.com/articles/monte-carlo-planning>.

—, **2017.** Release Planning A Monte Carlo Approach. *Google Drive*. [Online] 9. březen 2017. [Citace: 10. 12 2017.] <https://drive.google.com/file/d/0B6ugHLzjGmsqWWtrMG9rbXIGbkU/view>.

Dorda, Michal. 2017. Úvod do metody Monte Carlo. [Online] 2017. [Citace: 13. 12 2017.] http://homel.vsb.cz/~dor028/Aplikace_5.pdf.

Magennis, Troy. 2011. Introduction to Monte-Carlo Analysis for Software Development. *Slideshare*. [Online] 2011. [Citace: 24. 10 2017.] <https://www.slideshare.net/troymagennis/introduction-to-monte-carlo-analysis-for-software-development-troy-magennis-focused-objective>.

—, **2015.** The Economic Impact of Software Development Process Choice: Cycle-Time Analysis and Monte Carlo Simulation Results. *IEEE*. [Online] 2015. [Citace: 24. 10 2017.] <http://ieeexplore.ieee.org/document/7070420/>. ISBN 978-1-4799-7367-5.

Metropolis, N. 1987. The beginning of the Monte Carlo method. [Online] 1987. [Citace: 13. 12 2017.] <https://www.scribd.com/document/7107054/The-beginning-of-the-Monte-Carlo-method>.

Oxford Dictionaries. 2017. Definition of Monte Carlo method in English. [Online] 2017. [Citace: 13. 12. 2017.] https://en.oxforddictionaries.com/definition/monte_carlo_method.

Palisade. 2017. Monte Carlo Simulation. [Online] 2017. [Citace: 13. 12 2017.] http://www.palisade.com/risk/monte_carlo_simulation.asp.

Simulace.info. 2015. Monte Carlo method/cs. [Online] 2015. [Citace: 13. 12 2017.] http://www.simulace.info/index.php/Monte_Carlo_method/cs.

Vysoká škola báňská. 2017. Simulační metody typu Monte Carlo. [Online] 2017. [Citace: 13. 12 2017.] http://fast10.vsb.cz/krejsa/studium/sbs_tema03.pdf.