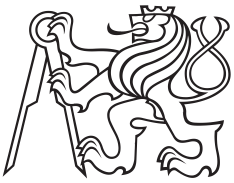
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Nástroj pro podporu výpočtu hodnoty Reálných opcí

Bc. Václav Trnka

Vedoucí práce: Ing. Pavel Náplava
Obor: Softwarové inženýrství
Studijní program: Otevřená Informatika
Květen 2017

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačů

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Trnka Václav

Studijní program: Otevřená informatika
Obor: Softwarové inženýrství

Název tématu: Nástroj pro podporu výpočtu hodnoty Reálných opcí

Pokyny pro vypracování:

1) Seznamte se s problematikou Reálných opcí, způsobu výpočtu jejich hodnoty a praktické aplikace, včetně základní rešerše existujících nástrojů.
2) V další práci se omezte již jen na výpočty hodnoty pomocí binomických stromů a navrhněte podpůrný nástroj pro ověřování různých způsobů výpočtu a hodnocení reálných opcí. Při návrhu se zaměřte na možnost flexibilně měnit jak vstupní parametry, tak i parametry mezivýpočtu.
3) Do návrhu zařaďte také metody pro různé průchody a změny vygenerovaných stromů.
4) Nezbytnou součástí návrhu musí být možnost uložit rozdělanou práci do souboru a generování vhodných grafických výstupů (reportů).
5) Navržený nástroj naimplementujte a ověřte na vybraných příkladech z oblasti IT investic (především srovnání investic do on-premise vs. cloud řešení).
6) Výsledky experimentálně a analyticky ověřte na datových sadách, vygenerovaných z příkladů publikovaných v literatuře.

Seznam odborné literatury:

[1] Starý, O.: Reálné opce. Praha: A plus, 2003.
[2] Scholleová, H.: Hodnota flexibility. Praha: C. H. Beck, 2007.
[3] Trigeorgis, L., Smit, Han T. J.: Strategic Investment, Real Options and Games, Princeton University Press, 2004.

Vedoucí: Ing. Pavel Náplava

Platnost zadání do konce letního semestru 2017/2018

prof. Dr. Michal Pěchouček, MSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 24.1.2017

Poděkování

Tímto bych rád poděkoval vedoucímu práce, Ing. Pavlu Náplavovi, za to, že se mi při vypracovávání této diplomové práce ve svém volném čase pravidelně věnoval a pomáhal mi. Dále bych chtěl poděkovat své rodině a přítelkyni, kteří mi byli oporou po celou dobu studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 19. 5. 2017

Abstrakt

Cílem této práce je usnadnit výpočet hodnot reálných opcí. Jejím základem je teorie popisující problematiku reálných opcí a problematiku IT investic. Z ní pak vychází praktická část, která ilustruje průběh návrhu, implementace a testování podpůrného experimentálního nástroje, který je výstupem této práce. Nástroj podporuje výpočty hodnot reálných opcí pomocí binomického modelu a pomocí zcela nové metodiky hodnocení IT investic.

Klíčová slova: Reálné opce, Aplikace, Nástroj, IT investice, Opce s pamětí, Cloud, On-premise, WPF, .NET, C#

Vedoucí práce: Ing. Pavel Náplava

Abstract

The aim of this thesis is to simplify the calculation of real option values. It is based on a theory describing the area of real options and IT investments. Theory is followed by practical part which describes the course of design, implementation and testing of the supporting experimental tool, which is the output of this work. The tool supports calculations of real option values using a binomial model and a completely new methodology for IT investments rating.

Keywords: Real Options, Application, Tool, IT investments, Option with Memory, Cloud, On-premise, WPF, .NET, C#

Title translation: Tool for Calculation of Real Option Values

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Hodnocení investic a to zejména pomocí reálných opcí je složitá činnost. Správně odhadnutý průběh investice přitom může ušetřit člověku mnoho času i peněz. Cílem této práce je seznámit se s danou problematikou a na jejím základě navrhnout, implementovat a na závěr také otestovat experimentální podpůrný nástroj, který by tyto výpočty usnadnil. Návrh se má soustředit hlavně na binomický model a dle něj generované binomické stromy. Nástroj má tyto stromy umět procházet, měnit a přehledně zobrazovat. Důraz je kladen také na oblast hodnocení IT investic a na srovnání cloudu vs. on-premise. Od těchto cílů se odvíjí struktura práce.

První kapitoly jsou věnované pojmům z oblasti informačních technologií, za nimi pak následují sekce týkající se pojmů z ekonomické praxe. V těchto kapitolách jsou vysvětleny pojmy jako finanční opce, reálné opce nebo binomický model. Jsou v nich popsané způsoby jejich výpočtu a dělení. Součástí práce jsou také dva ukázkové příklady z literatury, které výpočet přibližují. Dále pak následuje sekce popisující zcela novou metodiku hodnocení IT investic.

Následující kapitoly se pak již věnují samotnému aplikování získaných vědomostí. První kapitolu pomyslné praktické části tvoří analýza a návrh řešení. V rámci této kapitoly jsou analyzovány funkcionální i nefunkcionální požadavky, na jejichž základu je pak navržen cílový podpůrný nástroj. Další kapitola na návrh navazuje a ilustruje průběh implementace nástroje a jeho možnosti nasazení. Za těmito částmi pak následuje kapitola věnující se testování nástroje. Ta je rozdělená na testy týkající se správnosti výpočtu a na experimentální testování ilustrující zajímavé vlastnosti implementované metodiky. Závěrečnou kapitolu pak tvoří vyhodnocení, ve kterém je shrnuto naplnění cílů práce a popsán možný budoucí rozvoj aplikace.

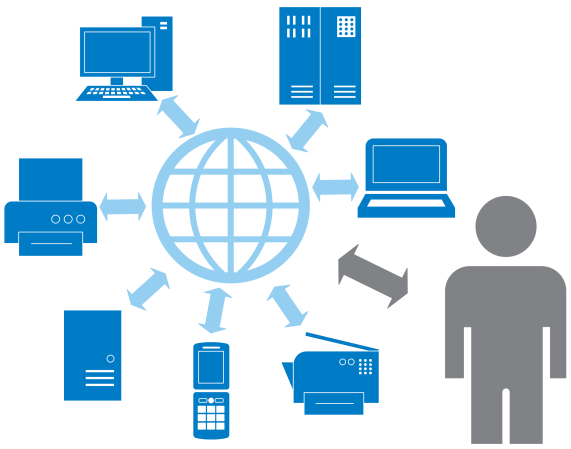
Motivací k výběru tohoto tématu byla možnost naučit se zajímavou technologii a podílet se na vývoji úplně nové metodiky týkající se hodnocení investic v oblasti informačních technologií.

Kapitola 2
Úvod do problematiky

Cílem této práce je především navrhnout, implementovat a otestovat nástroj pro podporu výpočtu hodnoty reálných opcí. Předtím, než bude možné přejít k navrhování požadovaného nástroje, je třeba definovat základní pojmy a uvést čtenáře do problematiky. Vzhledem k tomu, že se práce dotýká nejen opcí, ale také oblasti informačních technologií (IT), hned v úvodu této kapitoly jsou zavedené pojmy jako je Cloud computing nebo IT infrastruktura. Dále se se pak kapitola věnuje teorii týkající se hodnocení investic, a to konkrétně finančním opcím. Zakončená je jejich klasifikací, praktickou aplikací a způsoby výpočtu pomocí dvou nejznámějších modelů.

2.1 IT infrastruktura

Základním kamenem Cloud computingu a IT investic je beze sporu firemní IT infrastruktura. Co tento pojem ale vlastně znamená? Jedná se o souhrnný výraz, který zastřešuje nejen hardware, software, ale také komunikaci, služby a veškeré vybavení nutné k fungovaní firemních procesů. Propojí-li se tyto prvky a doplní navíc o lidský faktor, znalosti, zkušenosti a dovednosti, vznikne funkční celek zvaný IT infrastruktura (viz obrázek 2.1). [1][2]



Obrázek 2.1: IT infrastruktura.[autor]

2.2 Cloud computing

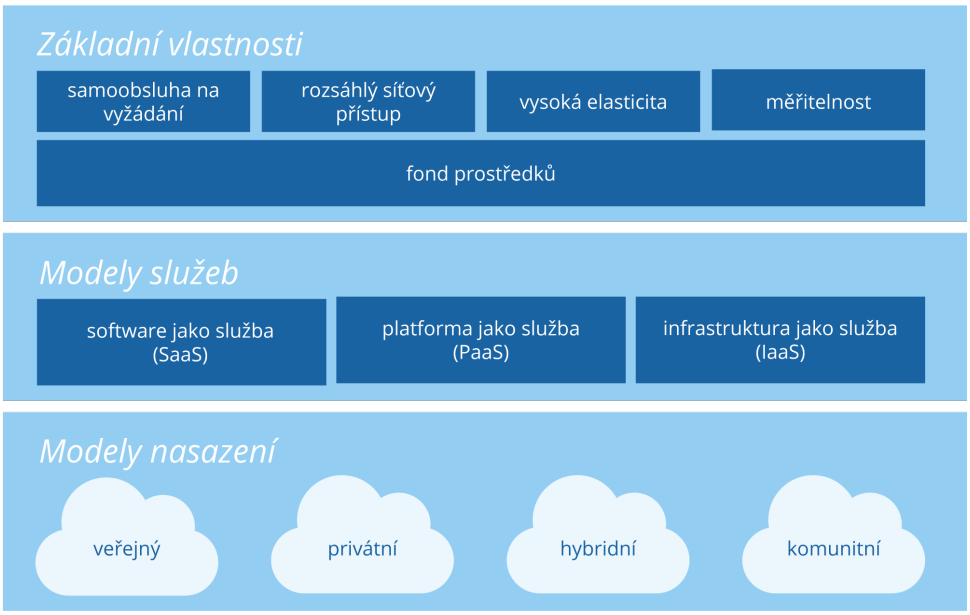
Pro Cloud computing existuje mnoho definic, které jsou si velmi podobné, není proto úplně jednoduché jeho význam shrnout na několik řádků. Poprvé se pojem objevil už v roce 1997 v přednášce Ramnatha Chellapa v americkém Dallasu. Myšlenka, dodávat výpočetní výkon podobně jako třeba elektřinu, je však mnohem starší, poprvé totiž zazněla už v roce 1961 z úst profesora Johna McCarthyho [3].

Obecně by se dalo říci, že Cloud computing je v nejjednodušším slova smyslu model, do kterého patří prakticky jakýkoliv program nebo služba dostupná přes síť, nejčastěji přes internet.

Oficiální definici poskytuje Národní institut standardů a technologie (NIST): *"Cloud computing je model, který na vyžádání umožňuje pohodlný a všudypřítomný síťový přístup ke sdílenému souboru konfigurovatelných výpočetních prostředků (např. sítě, servery, datová úložiště, aplikace a služby), které lze rapidně zajistit, nebo uvolnit a to s minimálním úsilím poskytovatele služeb. Tento cloudový model se skládá z pěti základních charakteristik, třech modelů služeb a čtyř modelů nasazení."* [4]

2.2.1 Model

NIST zavádí krom definice také rozdělení modelu, kterému se věnuje tato sekce. Toto rozdělení je ilustrováno obrázkem 2.2. Informace byly čerpány z oficiální publikace NIST [4] a z [2].



Obrázek 2.2: Model Cloud computingu dle NIST. [4]

Základní vlastnosti:

- *Samoobsluha na vyžádání* – Zákazník si může jednostranně a bez nutnosti interakce poskytovatele alokovat potřebný výpočetní výkon.
- *Rozsáhlý síťový přístup* – Prostředky jsou dostupné přes sít standardizovaným způsobem.
- *Vysoká elasticita* – Zdroje mohou být pružně zajišťovány i uvolňovány.
- *Měřitelnost* - Cloudové systémy automaticky kontrolují a optimalizují využívání zdrojů pomocí měření na úrovni abstrakce dané typem služby.
- *Fond prostředků* – Výpočetní prostředky poskytovatele slouží několika zákazníkům najednou. Fyzické i virtuální, dynamicky alokované zdroje jsou zákazníkům přiřazovány dle potřeby.

Modely služeb

- *Software jako služba (SaaS)* – Spotřebitel může používat aplikaci, kterou poskytovatel provozuje v cloudovém prostředí. Taková aplikace je dostupná vzdáleně přes sít prostřednictvím tenkého klienta (tedy přes klienta obsahující jen minimum logiky, například webový prohlížeč). Zákazník využívající SaaS se nestará o HW (hardware), licence, ani operační systém, v kterém aplikace běží. Pouze jí využívá jako službu.
- *Platforma jako služba (PaaS)* – Poskytovatel umožňuje spotřebiteli využívat prostřednictvím sítě veškeré prostředky nutné k vytváření a provozování vlastních aplikací v cloudovém prostředí. Zákazník tak má přístup k celé platformě a je jen na něm, jestli v ní bude provozovat vytvořený, nebo zakoupený program.
- *Infrastruktura jako služba (IaaS)* – Nejobecnější model, v rámci kterého spotřebitel získá výpočetní výkon, síť, úložiště na serveru a další esenciální prostředky nutné k tomu, aby mohl v získaném prostředí nasadit a provozovat libovolný SW (software) včetně operačních systémů a různých aplikací.

Modely nasazení

- *Veřejný* – Infrastruktura je určená pro širokou veřejnost. Model využívají nejrůznější organizace ze soukromého, akademického i státního sektoru.
- *Privátní* – Cloudová infrastruktura je exkluzivně určená pouze jedné organizaci, která může být složená z několika menších odběratelů (např. odděleních v jedné firmě).
- *Hybridní* – Tento vzor je kombinací dvou a více modelů nasazení. Tyto modely samy o sobě zůstávají samostatnými jednotkami, hybridní přístup však umožní jejich vzájemné propojení.

- *Komunitní* – Tento přístup vychází z privátního cloudu. Tuto variantu však využívá několik organizací se společnými zájmy tvořící komunitu.

2.2.2 Využívání

Využívání Cloud computingu jistě přináší spotřebiteli mnohé výhody, nese si však s sebou i několik nevýhod [5]:

Výhody

- *Minimální počáteční náklady* – Není třeba platit za HW ani za licence na SW. Obě tyto položky jsou přitom zejména v případě serverů velmi nákladné.
- *Schopnost efektivně snižovat a zvyšovat výkon* – Velkou výhodou Cloud computingu je možnost efektivně snižovat a zvyšovat pronajímanou výpočetní kapacitu. Spotřebitel je tak schopný pružně reagovat na zvýšenou nebo sníženou poptávku po jeho službě.
- *Zajištěná správa a údržba* – S cloudem odpadá potřeba zaměstnávat několik odborníků, kteří se o infrastrukturu starají. Systém je vždy funkční a aktualizovaný.
- *Rychlost nasazení* – Zprovoznění cloudové infrastruktury je otázka několika desítek minut.
- *Žádná migrace dat při přechodu na nové servery* – Migrace na nové servery často ve firmách probíhá hned po skončení servisní záruky. Tato pravidelná činnost v cloudu odpadá.
- *Zabezpečení na vysoké úrovni* – Zabezpečení cloudové infrastruktury bývá na velmi vysoké úrovni, na kterou by se spotřebitel svépomocí jen složitě dostával.
- *SLA* – Neboli Service Level Agreement je smluvně garantovaná dostupnost služby, často udávaná v procentech.

Nevýhody

- *Data jsou uložena v cizích datacentrech* – Data jsou sice šifrována, ale pořád zde existuje určité riziko, že se k nim dostane třetí strana. Zneužití dat přitom ani nemusí být úmyslné.
- *Nutnost připojení k síti* – Bez připojení k internetu nelze využívat prostředky umístěné ve vzdáleném datacentru.
- *Teoreticky pomalejší odezva* – V případě nestabilního internetového připojení hrozí pomalejší odezva, případně výpadek způsobený omezenou konektivitou do internetu.

- *Nutnost platit pravidelný poplatek* – Toto pro někoho může být i výhoda, každopádně je třeba počítat s tím, že poplatky za využívání cloudových služeb jsou pravidelné a to po celou dobu využívání služby.

- *Dlouhodobě hrozí vyšší ceny* – Z dlouhodobého hlediska může být Cloud computing dražším řešením. Je tomu tak zejména v případech, kdy HW bez problému funguje i po naplnění předpokládané životnosti.

Vyhodnocení

Jak je vidět výše, využívání Cloud computingu si s sebou nese výhody, ale i nevýhody. Pro účely této práce je nejzajímavější vlastnost cloudu možnost flexibilně měnit výkon. Aby bylo možné porovnat případné ušetřené náklady díky flexibilitě, je třeba se přesunout do oblasti financí, k čemuž slouží následující kapitoly.

2.3 Finanční opce a hodnocení investic

Po definování technických pojmů je třeba uvést čtenáře také do problematiky financí a hodnocení investic, do které spadá funkcionalita navrženého nástroje. V této části jsou na úvod popsané metody hodnocení investic a finanční opce. V druhé části je zpracovaná teorie týkající se stanovení hodnoty opce a v závěru podkapitoly jsou definované vzorce potřebné k výpočtům pomocí opcí.

2.3.1 Hodnocení investic

"Investici nelze chápat zúženě jako pouhé vydávání peněz, ale lze ji charakterizovat jako jednorázově (krátkodobě) vynaložené zdroje, které budou přinášet peněžní příjmy během delšího časového období." [6]

Investice lze tedy chápat jako výdaj, který se investorovi později vrátí. Tedy alespoň v případě, kdy se to vyplatí. K určení budoucího vývoje investice se používá mnoho ukazatelů. Jejich vhodnost a volba záleží na konkrétní situaci a na typu podkladového aktiva. Záleží také na fázi investičního procesu nebo cílech podniku. Při kvantitativním vyhodnocování se nejčastěji berou v potaz tři faktory – výnosnost, čas a míra rizika. Z nich vyplývající metody se dále dělí na *statické* a *dynamické*.

Statické metody ignorují faktor času a rizika a jsou vhodné hlavně pro zjištění orientačních ukazatelů jako je průměrný roční výnos, průměrný procentní výnos, nebo průměrná doba návratnosti. [6]

Dynamické metody čas zohledňují a pravidelně ho aktualizují formou diskontování všech vstupních parametrů použitých pro výpočet. Diskontováním je zde myšleno přepočtení budoucích výnosů v jednotlivých obdobích na současnou hodnotu a následné sečtení s použitím diskontní míry [7]. Mezi hlavní zástupce této kategorie dle [8][6] patří:

- *NPV (čistá současná hodnota)* – Základní a nejpoužívanější dynamická metoda, vhodná předně ke zjištění absolutního výnosu v penězích. Je použitelná univerzálně, dává srozumitelný výsledek a je jednoduchá na výpočet. Zohledňuje faktor času, rizika i časový průběh investice.
- *PI (index ziskovosti)* – Jedná se o poměr přínosů a počátečních kapitálových výdajů. Často se používá jako doplnění *NPV*. Je vhodný především k určení relativní výnosnosti, čím vyšší číslo index ukazuje, tím je projekt výhodnější.
- *IRR (vnitřní výnosové procento)* – Tato metoda je vhodná při preferenci určení relativního výnosu a možnosti investovat do většího portfolia. Lze jej chápat jako relativní výnos (rentabilitu), kterou projekt během svého života vykazuje.
- *PP (doba návratnosti)* – Doba návratnosti je vhodná při preferenci rychlé návratnosti. Je definována jako počet období, za které cash flow (tok výnosů) přinese hodnotu odpovídající počáteční investici.
- *DEVA (diskontovaná ekonomická přidaná hodnota)* – Tento ukazatel reprezentuje diskontovaný ekonomický zisk podniku, který ve firmě zůstane po uspokojení všech poskytovatelů kapitálu. Je vhodný při preferenci absolutního výnosu.

Všechny tyto ukazatelé se pro hodnocení investic využívají a při vhodném zvolení dávají relevantní výsledky. Mají však jednu společnou nevýhodu, kvůli které nejsou pro tuto práci vhodné. Vůbec totiž neuvažují možnost měnit již přijatá rozhodnutí. Základní výhodou Cloud computingu je přitom možnost flexibilně reagovat na vzrůstající, nebo klesající poptávku a tedy i měnit svá rozhodnutí o počáteční konfiguraci. Pro hodnocení IT investic je proto třeba zvolit jinou metodiku a tou jsou finanční, respektive reálné opce.

2.3.2 Finanční opce

Cílem této práce je navrhnout aplikaci podporující výpočet hodnot reálných opcí. Jak ale čtenář zjistí v této kapitole, pro pochopení reálných opcí je nutné nejprve znát základní pojmy z domény finančních opcí. Obě problematiky jsou totiž úzce provázané a jedna vychází z druhé. Protože je téma opcí velmi rozsáhlé, jsou v této práci zavedeny potřebné pojmy pouze velmi okrajově, pro hlubší pochopení problematiky lze čerpat z [6][9][10].

Doktorka Scholleová definuje finanční opce takto:

"Při uzavření finanční opce získává kupující právo, nikoli však povinnost, koupit (opce na koupi - call) či prodat (opce na prodej - put) podkladové aktivum za předem stanovenou cenu (realizační cena, expirační cena, expiration price, X) v předem daném termínu (životnost opce, doba vypršení opce, doba splatnosti, time to maturity, T)." [6]

Právo prodat, nebo naopak koupit, má kupující na základě jednostranně výhodné smlouvy, kterou uzavírá kupující a prodávající. Poplatek za toto

právo se nazývá prémie. V případě, že se držitel opce rozhodne svého práva vzdát a opci nevyužít, prémie zůstává druhé straně. Držitel opce má přitom ztrátu pouze ve výši zaplacené prémie. Předmětem smlouvy je tzv. podkladové aktivum, od kterého se obchod odvozuje. [9]

2.3.3 Klasifikace opcí

V této části se čtenář dozví o základním dělení opcí, které je aplikovatelné jak na finanční opce, tak na opce reálné. Informace pro tuto část pochází z literatury [6] a [9].

Dle typu opce

- *Kupní (call)* – Kupující má právo koupit podkladové aktivum za stanovenou cenu.
- *Prodejní (put)* – Prodávající má právo prodat podkladové aktivum za dohodnutou cenu.

Dle pozice

- *Krátká (short)* – V této pozici je strana, která se zavázala ke splnění povinnosti. Jedná se o nevýhodnou pozici.
- *Dlouhá (long)* – Zde je naopak strana, která zaplatila za právo se rozhodnout. Jde o výhodnou pozici.

Dle doby

- *Evropská* – Opce může být uplatněna pouze v den expirace.
- *Americká* – Opce může být uplatněna kdykoliv od upsání opce až do doby expirace.

Dle vztahu současné a expirační ceny

Toto dělení závisí na vztahu spotové ceně S a expirační (neboli realizační) ceně X. Spotová cena reprezentuje aktuální hodnotu podkladového aktiva, expirační cena představuje dohodnutou cenu při využití opčního práva.

- *V penězích (in the money)* – Situace, kdy je výhodné opci využít, tedy když spotová cena převyšuje expirační cenu pro kupní opci, respektive když expirační cena převyšuje spotovou pro prodejní opci.
- *Mimo peníze (out of the money)* – Případ, kdy se využití opce nevyplatí, tedy když je spotová cena nižší než expirační cena pro kupní opci, respektive když spotová cena převyšuje expirační cenu pro prodejní opci.
- *Na penězích (at the money)* – Pokud je opce tzv. na penězích, znamená to, že se $S = X$, jinak řečeno nezáleží na tom, zda držitel opční právo využije, nebo ne.

2.3.4 Hodnota opce

Opce, jakožto právo na výhodné postavení, má svojí hodnotu. Tu lze vyjádřit jako složení její vnitřní a časové hodnoty. Vnitřní hodnota přitom odráží vztah aktuální a realizační ceny, čímž definuje výši zisku při okamžitém využití opce. Vnitřní hodnota je rovna nule, pokud je rozdíl spotové a realizační ceny pro držitele opce nevýhodný. V opačném případě je rovna výši zisku bez zahrnutí opční prémie. [6][9]

Časová hodnota reflektuje momentální vliv nabídky a poptávky po dané opci na trhu. V jednoduchosti se jedná o částku, kterou je ochoten zaplatit kupující prodávajícímu za to, že se během doby do vypršení opce příznivě změní podmínky na trhu. Se zkracující se dobou vypršení opce tak časová hodnota klesá, až v den její splatnosti splyne s vnitřní hodnotou. Je to způsobené tím, že se s dobou do vypršení opce snižuje i pravděpodobnost pozitivní změny hodnoty podkladového aktiva. [9]

Faktory ovlivňující hodnotu a jejich značení

Hodnota opce je přímo ovlivňována několika charakterizačními faktory, které se týkají hodnoty podkladového aktiva, podmínek uzavřené opční smlouvy a ekonomické situace okolí. Jedná se především o [6][9]:

- S – *Cena podkladového aktiva (spotová, okamžitá cena)* – Jedná-li se o kupní opci (call) a hodnota aktiva vzroste, musí zákonitě vzrůst i hodnota celé opce. Pokud naopak aktivum zlevní, potom zlevní i call opce. Analogicky je to u prodejní opce (put). Při poklesu ceny aktiva hodnota opce roste, při růstu naopak klesá. V obou případech jsou výkyvy způsobené změnou potenciálního zisku pro držitele opce při jejím využití.
- X – *Realizační cena (expirační cena)* – Čím více převyšuje u prodejní opce realizační cena spotovou, tím výhodnější využití opce pro držitele je. U kupní opce je to naopak, vyšší spotová cena znamená vyšší zisk. Držitel opce totiž může nakoupit za nižší realizační cenu a obratem prodat za vyšší cenu spotovou.
- T – *Čas zbývající do expirace (doba do vypršení)* – Čím delší doba zbývá do vypršení opce, tím větší je pravděpodobnost pozitivní změny ceny aktiva a tím vyšší je hodnota opce. Doba do vypršení opce je hlavním faktorem ovlivňující její časovou hodnotu. Jednotky T jsou standardně roky.
- r – *Bezriziková úroková míra (risk free rate)* – S růstem r je kupní opce hodnotnější, protože současně roste hodnota podkladového aktiva. U prodejní opce je to ze stejných důvodů naopak, s rostoucí bezrizikovou úrokovou mírou hodnota klesá. Stanovuje se z úrokových měr státních dluhopisů.
- σ *nebo* σ^2 – *Míra volatility (kolísavost podkladového aktiva)* – Tento faktor vychází ze stálosti hodnoty podkladového aktiva. Čím většími

výkyvy se aktivum potýká, tím vyšší je kolísavost (někdy také rizikovost) a tím vyšší musí být také hodnota opce. Může se totiž snadno stát, že finální cena se bude výrazně lišit od původně očekávané hodnoty a to jak pozitivně, tak i negativně. To může prudce ovlivnit budoucí zisk, nebo ztrátu. Záruka realizační ceny poskytovaná opcí se tak musí projevit na její hodnotě. Míra volatility se charakterizuje pomocí rozptylu σ^2 nebo pomocí směrodatné odchylky σ.

2.4 Způsoby výpočtu hodnoty opcí

V předchozí části se čtenář dozvěděl o finančních opcích a o jejich hodnotě. V této části se seznámí s tím, jak lze tyto hodnoty prakticky počítat. Pro stanovení hodnoty finančních i reálných opcí existuje řada modelů. Nejdůležitější jsou ale dva základní – binomický model a spojitý model, jejichž způsobu výpočtu je věnovaná tato sekce.

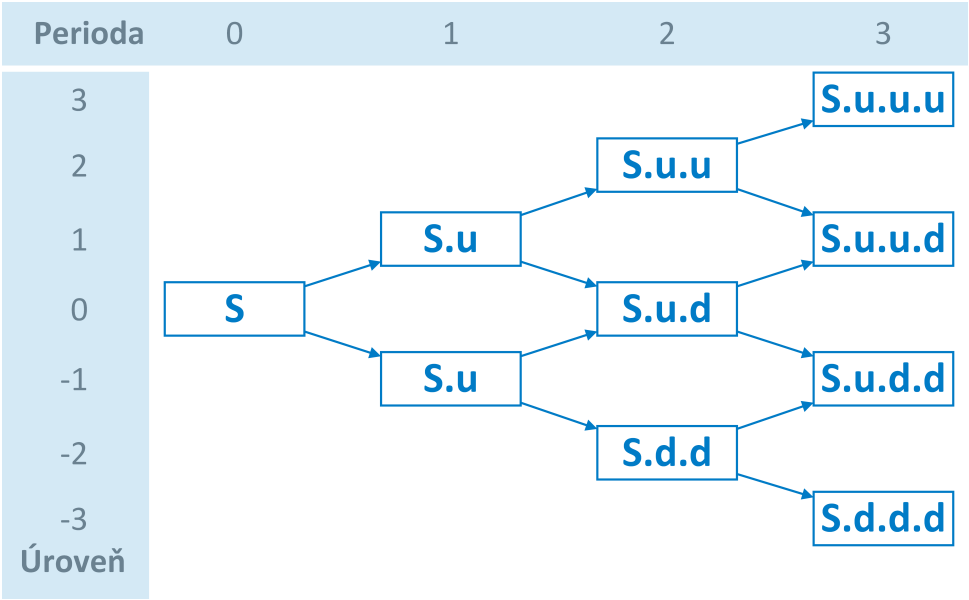
2.4.1 Binomický model

V této podkapitole je definovaný binomický model pro stanovení hodnoty opce, který je pro tuto práci stěžejní. Vzorce v této sekci jsou jak úvodem do problematiky, tak reálným předpisem implementované funkcionality.

Binomický model je nespojitý. Vychází z toho, že jde vývoj opce během její životnosti (T) rozdělit do konečného počtu stejně dlouhých období (značeno n), během nichž buď dochází k růstu s indexem u a pravděpodobností p, nebo k poklesu s indexem d a doplňkovou pravděpodobností $(1-p)$. Předpokládá se tedy, že se cena aktiva mění pouze v diskrétních časových okamžicích. Příklady v literatuře často obsahují hodnoty, pro které platí, že $T = n$. To znamená, že jedno období odpovídá jednomu roku. Jako zdroje pro popis modelu byly použity práce [6], [9] a [11].

Aby bylo možné binomický model použít, je třeba splňovat jeho předpoklady. Konkrétně se očekává, že nelze dosáhnout bezrizikového zisku, tedy že neexistuje možnost arbitráže. Další podmínkou je také platnost zákona jedné ceny. To znamená, že pokud mají dvě různá aktiva v budoucnu stejnou cenu, musí jí mít i dnes. Dále počítá s existencí dokonalých trhů, kde neexistují transakční náklady, daně ani omezení na krátký prodej. Podkladové aktivum je tak nekonečně dělitelné. Také by mělo platit, že výnos libovolného aktiva je roven bezrizikové sazbě.

Myšlenka modelu stojí na tom, že cena podkladového aktiva S může v čase T_i nabývat pouze dvou různých diskrétních hodnot a to buď S_1^u, pokud jeho cena vzrostla, nebo S_1^d, pokud cena klesla. Pravděpodobnost růstu a tedy i doplňková pravděpodobnost poklesu se přitom v čase nemění. Předpokládaný vývoj spotové ceny podkladového aktiva dle modelu ilustruje obrázek 2.3.



Obrázek 2.3: Předpokládaný vývoj ceny podkladového aktiva dle binomického modelu. [autor]

V níže uvedených vzorcích jsou použity tyto parametry:

- u – index růstu,
- d – index poklesu,
- e – základ přirozeného logaritmu,
- S – spotová cena,
- X – realizační cena,
- r – bezriziková úroková míra,
- T – čas zbývající do expirace,
- n – počet období v čase zbývajícím do expirace,
- σ – míra volatility vyjádřená směrodatnou odchylkou,
- p – pravděpodobnost růstu,
- $(1 - p)$ – pravděpodobnost poklesu,
- C_u – vnitřní hodnota kupní opce (call) na konci 1. období pro případ růstu,
- C_d – vnitřní hodnota kupní opce na konci 1. období pro případ poklesu,
- P_u – vnitřní hodnota prodejní opce (put) na konci 1. období pro případ růstu,

- P_d – vnitřní hodnota prodejní opce na konci 1. období pro případ poklesu,
- C – hodnota evropské kupní opce,
- P – hodnota evropské prodejní opce,
- $(C_i)_n$ – i-tá vnitřní hodnota americké kupní opce v období n,
- $(P_i)_n$ – i-tá vnitřní hodnota americké prodejní opce v období n.

Aby bylo možné vývoj spočítat, je třeba nejprve znát hodnotu u a d. Pro ty platí:

$$u = e^{\sigma\sqrt{\frac{T}{n}}}$$
$$d = e^{-\sigma\sqrt{\frac{T}{n}}}$$

Protože pokles a následný růst musí vyústit v původní hodnotu, musí platit také:

$$u \cdot d = e^{\sigma\sqrt{\frac{T}{n}}} \cdot e^{-\sigma\sqrt{\frac{T}{n}}} = e^0 = 1$$

Pro výpočet hodnoty opce je dále nutné znát pravděpodobnost růstu p a pravděpodobnost poklesu $(1-p)$:

$$p = \frac{(1+r)^{\frac{T}{n}} - d}{u - d}$$
$$1 - p = \frac{(1+r)^{\frac{T}{n}} - u}{d - u}$$

Pro které platí, že:

$$p + (1-p) = 1$$

Aby bylo výhodné opci uplatnit, musí být její vnitřní hodnota větší než nula. V opačném případě je její hodnota nulová, protože opce nebude uplatněna. Proto se ve vzorcích využívá funkce $max(x, 0)$, která vrací 0, pokud by bylo x záporné.

Pro vnitřní hodnotu kupní opce na konci prvního období platí:

$$C_u = max(u \cdot S - X, 0)$$
$$C_d = max(d \cdot S - X, 0)$$

Pro výpočet vnitřní hodnoty prodejní opce na konci prvního období je třeba prohodit S a X :

$$P_u = max(X - S \cdot u, 0)$$
$$P_d = max(X - S \cdot d, 0)$$

Pro hodnotu evropské kupní opce pro n období pak platí:

$$C = \frac{1}{(1+r)^n} \cdot \sum_{i=0}^{n} \frac{n!}{i! \cdot (n-i)!} \cdot p^i \cdot (1-p)^{n-i} \cdot max(S \cdot u^i \cdot d^{n-i} - X, 0)$$

Analogicky pro hodnotu evropské prodejní opce pro n období platí:

$$P = \frac{1}{(1+r)^n} \cdot \sum_{i=0}^{n} \frac{n!}{i! \cdot (n-i)!} \cdot p^i \cdot (1-p)^{n-i} \cdot max(X - S \cdot u^i \cdot d^{n-i}, 0)$$

Pro účely této práce je však zajímavější výpočet hodnoty americké opce. Aby jí bylo možné určit, je třeba rekurentně postupovat binomickým stromem zpět, kde i-tá hodnota (pro $i = 1 \dots n$) uzlu v období n je rovna vnitřní hodnotě kupní opce:

$$(C_i)_n = max(S \cdot u^i \cdot d^{n-i} - X, 0)$$

a i-tá hodnota uzlu v periodě $n-1$ se pak pro americkou kupní opci počítá dle vzorce:

$$(C_i)_{n-1} = max\left[\frac{1}{(1+r)^{T/n}} \cdot (p \cdot (C_i)_n + (1-p) \cdot (C_{i+1})_n), max((S_i)_{n-1} - X, 0)\right]$$

Stejný princip platí také pro vnitřní hodnotu americké prodejní opce:

$$(P_i)_n = max(X - S \cdot u^i \cdot d^{n-i}, 0)$$

a i-tá hodnota uzlu v periodě $n-1$ se pak pro americkou prodejní opci počítá dle vzorce:

$$(P_i)_{n-1} = max\left[\frac{1}{(1+r)^{T/n}} \cdot (p \cdot (P_i)_n + (1-p) \cdot (P_{i+1})_n), max(X - (S_i)_{n-1}, 0)\right]$$

Velkou výhodou binomického modelu je možnost použít ho jak pro evropské opce, tak pro ty americké. Je také velmi snadno aplikovatelný na opce reálné. Obecně platí, že čím je zvolen větší počet intervalů (n) do vypršení opce (T), tím přesnější průběh model zobrazuje. Nevýhodou je naopak jeho velká závislost na stanovení indexů u a d. Sice je pro ně pevně daný vzorec, ale pokud jsou do něj dosazené nevhodné hodnoty, může být výsledek velmi nepřesný. Úvaha, že v daný moment může dojít pouze k růstu, nebo k poklesu je také určité zjednodušení. Přesto se binomický model používá a v jistém smyslu je stejně přesný a kvalitní jako níže uvedený Black-Scholesův model. Ten je totiž zároveň limitním případem binomického modelu. [9][6]

V rámci binomického modelu je dále možné opce řetězit, tedy lze z poslední periody stromu sestavit nový strom, který reprezentuje další opci. Tento nový podstrom je na původní opci nezávislý. Jeho přidání se projeví pouze při rekurentním zpětném průchodu stromem, díky kterému se hodnoty z posledního stromu promítnou až do kořene původní opce (průchod začne v poslední periodě nového stromu a postupuje až do stromu původního). Tato funkcionalita byla součástí požadavků na navrhovaný nástroj.

2.4.2 Black-Scholesův spojitý model

Druhý přístup k oceňování opcí poskytuje Black-Scholesův spojitý model. Ten předpokládá, že lze časový úsek rozdělit na nekonečně mnoho malých podúseků, které tvoří spojitou křivku. Základním předpokladem je tedy změna ceny.

Aby ale bylo možné spojitý model použít, je třeba splnit další povinné náležitosti. Vyhodnocovaná opce musí být evropského typu, na americké opce vzorec aplikovat nelze, což je jeho velké mínus. Podobně jako binomický model vyžaduje existenci dokonalých trhů zanedbávající vliv daní a transakčních nákladů. Aktiva jsou na takovém trhu nekonečně dělitelná a neexistují omezení jejich nákupů ani prodejů. Veškeré dostupné informace jsou zahrnuty v cenách, jakožto v jediných nositelích těchto informací. Dále se počítá s tím, že střední hodnota výnosu podkladového aktiva a její směrodatná odchylka jsou v čase konstantní, z něj nedochází k vyplácení dividend a také na trhu lze bez omezení půjčovat peníze. Bezriziková úroková míra je v modelu konstantní, a navíc stejná pro všechny doby splatnosti. Stejně jako v binomickém modelu není možná arbitráž.

Informace pro tuto sekci pochází z literatury [12] a [6], využívány jsou zde tyto parametry:

- S – aktuální cena aktiva,
- X – realizační cena aktiva,
- r – bezriziková úroková míra,
- T – doba do vypršení opce,
- σ – směrodatná odchylka ceny aktiva,
- e – základ přirozeného logaritmu,
- $N(d_1), N(d_2)$ – hodnoty distribuční funkce normálního rozdělení pro d_1, d_2,
- C – hodnota evropské kupní opce (call),
- P – hodnota evropské prodejní opce (put).

Hodnota evropské kupní (call) opce v době T lze do její splatnosti určit dle vzorce:

$$C = S \cdot N(d_1) - X \cdot e^{-rT} \cdot N(d_2)$$

Pro hodnotu prodejní opce pak platí:

$$P = -S \cdot N(-d_1) + X \cdot e^{-rT} \cdot N(-d_2)$$

kde pro d_1 a d_2 platí:

$$d_1 = \frac{ln(\frac{S}{X}) + (r + \frac{\sigma^2}{2}) \cdot T}{\sigma \cdot \sqrt{T}}$$

$$d_2 = d_1 - \sigma \cdot \sqrt{T} = \frac{ln(\frac{S}{X}) + (r - \frac{\sigma^2}{2}) \cdot T}{\sigma \cdot \sqrt{T}}$$

Největší výhodou spojitého modelu je možnost stanovit hodnotu opce kdykoliv. Mezi jeho nevýhody patří podobně jako u binomického modelu obtížnost určení míry volatility a jak už bylo řečeno, lze ho aplikovat pouze na opce evropského typu. Je tedy pro účely této práce nevhodný a dále v ní nefiguruje.

Kapitola 3
Reálné opce

Po objasnění potřebných vzorců je možné přejít od finančních opcí k reálným. Na úvod se v této kapitole čtenář dozví, co to reálné opce jsou, následně se seznámí s jejich dělením a s ukázkovými příklady. Součástí kapitoly je také speciální případ aplikace úplně nové opční metodiky týkající se IT investic. Kapitola je pak zakončená základním souhrnem nástrojů využívaných ke stanovení hodnoty reálných opcí.

3.1 Popis

Díky podobnosti finančního odvětví s dalšími odvětvími a pro potřebu hodnocení nejen finančních investic vznikly opce reálné. V oblasti finančních opcí se sice uvažuje o možnosti prodat, nebo koupit podkladové aktivum za předem stanovených podmínek, *jako opci lze ale chápat téměř jakoukoliv situaci v podniku, která pracuje s rozhodováním na základě podnětu změny vnějšího okolí a jeho vlivu na další vývoj firmy, tj. hlavně rozhodování o akcích investičních. Obdobně, jako jsou finanční opce chápány jako právo na budoucí nákup nebo prodej nějakého aktiva, tak reálné opce můžeme chápat jako právo na inkasování budoucích peněžních toků souvisejících s realizací nějakého projektu.* [9]

Reálné opční metody nelze vnímat jako konkurenci skupiny dynamických metod, ale jako vhodný doplněk, který pracuje s rizikem a ohodnocením flexibility v podniku. Jako opci lze chápat téměř každou možnost vytvoření situace, která dává subjektu právo pozdějšího rozhodování. [6]

Příkladem reálné opce je např. zařízení, které umí flexibilně zpracovávat určité suroviny. Na základě aktuální situace na trhu je pak možné změnit způsob zpracování a efektivně tak uspokojit poptávku. Z pohledu IT je reálnou opcí třeba zakoupení serveru s volným slotem na paměť. Majitel si koupí nechává možnost v budoucnu paměť rozšířit, nebo nechat server tak, jak ho koupil. Dalším příkladem je licence zakoupená na určitou dobu. Po vypršení licence se podnik může rozhodnout, zda licenci prodlouží, nebo jí nechá propadnout, a může se tak soustředit na jinou činnost. Reálnou opcí může být i investice do výzkumu a vývoje. Ta později podniku umožní rozhodnout se, zda s vývojem pokračovat, nebo nikoliv. [12]

3.2 Klasifikace

Reálné opce lze rozdělit do mnoha kategorií. Jednoznačně zařadit konkrétní projekt do té, či oné kategorie už ale tak jednoduché není. Projekty totiž můžou být natolik propojené, že je zařadit jednoduše nelze. V nejobecnější rovině lze reálné opce rozdělit do třech kategorií s několika podtypy, což bylo provedeno v této sekci. Každý podtyp je pak zařazený do opční kategorie z části 2.3.3, což ukazuje, že po stanovení všech parametrů je možné vypočítat hodnotu reálných opcí stejným způsobem jako hodnotu opcí finančních (viz 2.4). Fakta pochází z [6][12] a [11].

3.2.1 Opce učení

Principem opcí učení je odsunout rozhodnutí až do zjištění rizikových faktorů. Využívá se tak zejména v období před investicí. Pokud se vývoj ukazuje jako nevýhodný, opce není využita a propadá. Do kategorie spadá hlavně opce vyčkávání a opce rozfázování.

- *Opce vyčkávání* – Tato opce dává držiteli právo odložit rozhodnutí do doby, než bude známo více informací o projektu. Příkladem opce odložení je nevyužitá parcela držená jako realita pro případný budoucí rozvoj.

  Opce vyčkávání je kupní opcí, většinou amerického typu. Investovaná částka představuje realizační cenu opce, cena spotová je reprezentována současnou hodnotou budoucích cash flow plynoucích z investice. Doba do expirace je čas, na který lze investiční akci odložit.

- *Opce rozfázování* – Rozfázování dává držiteli právo postupně vynakládat investiční výdaje. Současně mu ale dává také možnost opustit projekt při nepříznivém vývoji. Opce je vhodná hlavně pro projekty s malým výnosem a vysokou nejistotou.

  Formálně se jedná o složenou opci z několika kupních opcí, přičemž předchozí opce vždy vstupuje do té následující. Expirační cena se rovná investici při vstupu do fáze. Současnou cenu poslední opce tvoří současná hodnota cash flow v daném období, u ostatních opcí je to hodnota následující opce. Dobu expirace představuje délka jedné fáze.

3.2.2 Růstové opce

Tyto opce jsou vhodné k využití v rámci investiční fáze. Hodnotu investice tvoří budoucí úspěšné investiční možnosti, na které je možné navázat. Příležitosti jsou čerpány z předchozí investice, rozhodování však už probíhá na počátku investice budoucí.

- *Opce inovace* – Někdy označovaná také jako růstová opce dává držiteli právo na hodnotu následujících projektů za akceptace již proběhlých projektů. Jedná se o strategická rozhodnutí o dlouhodobých záležitostech podniku pro která platí, že od nich může management kdykoliv upustit.

Jedná se o americkou kupní opci, nebo složení několika kupních opcí (compound opce). Používá se zejména v oblasti technologií a v poli náročném na vývoj.

- *Opce rozšíření* – Právo na expanzi umožňuje změnit rozsah projektu a přizpůsobit tak rozsah produkce aktuální situaci na trhu. Jejím opakem je opce zúžení projektu, která se využívá při nepříznivém vývoji na trhu.

  Z pohledu kategorizace jde o kupní opci na budoucí cash flow za cenu investičních výdajů pro rozšíření projektu, nebo o prodejní opci za cenu budoucích propadlých cash flow v případě zúžení projektu.

3.2.3 Opce zajištění

Opce zajištění se využívají v období nepříznivého vývoje situace na trhu v období během a po investici. Jejich cílem je minimalizovat ztrátu.

- *Opce záměny* – Jinak také opce flexibility dává právo volného pohybu mezi volbou vstupů. Soudě podle vývoje aktuální situace na trhu je možné flexibilně měnit vstupy i výstupy. Flexibilita je přitom zabudována přímo v aktivech, která mohou v různých situacích přinášet různé výhody. Příkladem může být třeba zaměstnanec zaučený na více pozicích.

  Opce záměny se počítá jako kombinace americké prodejní opce (právo na ukončení využívání jednoho vstupu) s americkou kupní opcí (právo začít využívat jiný vstup). Své využití najde především v energetickém a chemickém průmyslu.

- *Opce přerušení* – Právo na přerušení umožňuje dočasně přerušit činnost při nepříznivé situaci na trhu. Využívá se v situacích, kdy by příjmy nepokryly ani náklady spojené s výrobou.

  Jedná se přitom o kupní opci, která dává právo získat budoucí cash flow za cenu uhrazení nákladů na výrobu v době přerušení.

- *Opce ukončení* – Tato opce dává svému držiteli právo na předčasné ukončení projektu a rozprodání s ním souvisejících aktiv za účelem minimalizace ztrát.

  Po formální stránce je to americká prodejní opce na hodnotu projektu s realizační cenou ve výši zůstatkové ceny aktiv. Současnou cenu tvoří hodnota budoucích cash flow, které by projekt mohl generovat.

Jak se čtenář přesvědčil v této sekci, reálné opce opravdu vychází z opcí finančních. Za předpokladu správného určení hodnot je tedy možné plně aplikovat popsané modely pro stanovení hodnoty opce z části 2.4 také na reálné opce. V následující sekci jsou pro ukázku a potvrzení tohoto tvrzení uvedené příklady z literatury [6] pro jednu opci na rozšíření projektu a jednu opci na zúžení projektu.

3.3 Příklady výpočtů pomocí reálných opcí

Tato sekce obsahuje dva příklady z literatury [6]. Jejím cílem je přiblížit čtenáři způsob výpočtu hodnoty reálné opce pomocí binomického stromu, oba příklady jsou mimo jiné také referenčními daty implementované aplikace.

3.3.1 Opce na rozšíření projektu

Jak již bylo řečeno, jedná se o americkou kupní opci s těmito parametry:

- S – současná cena odpovídá hodnotě cash flow rozšířené části projektu,
- X – realizační cena jsou dodatečné investiční výdaje pro rozšíření základního projektu,
- T – doba životnosti opce je doba, po kterou může být rozšíření uplatněno,
- σ^2 – volatilita hodnoty budoucích cash flow,
- r – bezriziková úroková míra.

Příklad 5.2 z [6]:
Firma při budování objektu pro realizaci projektu ponechala volnou plochu s ohledem na možnost rozšíření kapacity kdykoli během 6leté životnosti projektu. Rychlé dovybavení vyžaduje dodatečné investování 100 000 Kč a přineslo by dodatečné cash flow 200 000 Kč. Rozšíření se jeví na první pohled jako výhodné, ale zisk je závislý na existenci dodatečné poptávky, jejíž volatilita byla odhadnuta na 0,4 (rozptyl). Jaká je hodnota flexibility z možnosti využití volné kapacity? (Bezriziková úroková míra je 0,045.)
Řešení:

$$\begin{gathered} S = 200\ 000\ \text{Kč} \\ X = 100\ 000\ \text{Kč} \\ T = 6\ \text{let} \\ n = 6 \\ \sigma^2 = 0{,}4 \\ r = 0{,}045 \end{gathered}$$

Dopočtené parametry:

$$\begin{gathered} u = 1{,}88 \\ d = 0{,}53 \\ p = 0{,}38 \\ 1 - p = 0.62 \end{gathered}$$

Příklad lze řešit binomickým modelem. Z těchto parametrů je pak nejprve nutné spočítat vývoj předpokládaných cen podkladového aktiva (obrázek 3.1), z něj určit vývoj vnitřních hodnot (obrázek 3.2), a následně zpětným průchodem dopočítat hodnotu opce (obrázek 3.3). Výsledkem je hodnota

v kořenu stromu, která říká, že hodnota pouhého práva na rozšíření investice je 151 529 Kč.

	perioda 0	perioda 1	perioda 2	perioda 3	perioda 4	perioda 5	perioda 6
úroveň 6							8 893 274
úroveň 5						4 724 869	
úroveň 4					2 510 255		2 510 255
úroveň 3				1 333 662		1 333 662	
úroveň 2			708 556		708 556		708 556
úroveň 1		376 445		376 445		376 445	
úroveň 0	200 000		200 000		200 000		200 000
úroveň -1		106 257		106 257		106 257	
úroveň -2			56 453		56 453		56 453
úroveň -3				29 993		29 993	
úroveň -4					15 935		15 935
úroveň -5						8 466	
úroveň -6							4 498

Obrázek 3.1: Předpokládaný vývoj hodnoty podkladového aktiva kupní opce. [6]

	perioda 0	perioda 1	perioda 2	perioda 3	perioda 4	perioda 5	perioda 6
úroveň 6							8 793 274
úroveň 5						4 624 869	
úroveň 4					2 410 255		2 410 255
úroveň 3				1 233 662		1 233 662	
úroveň 2			608 556		608 556		608 556
úroveň 1		276 445		276 445		276 445	
úroveň 0	100 000		100 000		100 000		100 000
úroveň -1		6 257		6 257		6 257	
úroveň -2			0		0		0
úroveň -3				0		0	
úroveň -4					0		0
úroveň -5						0	
úroveň -6							0

Obrázek 3.2: Předpokládaný vývoj vnitřních hodnot kupní opce. [6]

	perioda 0	perioda 1	perioda 2	perioda 3	perioda 4	perioda 5	perioda 6
úroveň 6							8 793 274
úroveň 5						4 629 175	
úroveň 4					2 418 682		2 410 255
úroveň 3				1 246 033		1 237 968	
úroveň 2			630 086		616 983		608 556
úroveň 1		312 168		297 899		280 752	
úroveň 0	**151 529**		139 763		123 743		100 000
úroveň -1		63 965		52 882		36 389	
úroveň -2			22 101		13 242		0
úroveň -3				4 818		0	
úroveň -4					0		0
úroveň -5						0	
úroveň -6							0

Obrázek 3.3: Zpětný přepočet hodnoty americké kupní opce. [6]

3.3.2 Opce na zúžení projektu

Opce na zúžení je analogií k opci rozšíření umožňující snížit velikost projektu rozprodáním části výrobních kapacit. Počítá se jako americké prodejní opce.

- S – současná cena odpovídá hodnotě cash flow likvidované části projektu,
- X – realizační cena jsou uspořené investiční výdaje,
- T – doba životnosti opce je doba, po kterou může být zúžení uplatněno,
- σ^2 – volatilita hodnoty budoucích cash flow,
- r – bezriziková úroková míra.

Příklad 5.3 z [6]:
Podnik, jehož hodnota je 50 mil. Kč, může v případě nepříznivě se vyvíjející situace zúžit kdykoli v průběhu příštích 5 let svou výrobu až o 20 % a tím ušetřit náklady ve výši 8 mil. Kč (volatilita v odvětví vyjádřená rozptylem je 0,6, bezriziková úroková míra 3,8 %). Jaká je hodnota takové flexibility?
Řešení:

$$\begin{gathered} S = 10\ 000\ 000 \text{ Kč} \\ X = 8\ 000\ 000 \text{ Kč} \\ T = 5 \text{ let} \\ n = 5 \\ \sigma^2 = 0{,}6 \\ r = 0{,}038 \end{gathered}$$

Dopočtené parametry:

$$\begin{gathered} u = 2{,}17 \\ d = 0{,}46 \\ p = 0{,}34 \\ 1 - p = 0{,}66 \end{gathered}$$

I tento příklad lze řešit binomickým stromem. Ze zadaných a dopočtených parametrů je nejprve třeba spočítat vývoj hodnoty obětované firmy (obrázek 3.4). Z něj je pak nutné odvodit vývoj vnitřní hodnoty opce (obrázek 3.5). A na závěr provést zpětný přepočet (obrázek 3.6). Výsledek je opět uvedený v kořenu stromu. V tomto případě má právo na zúžení hodnotu 3 873 tis. Kč.

	perioda 0	perioda 1	perioda 2	perioda 3	perioda 4	perioda 5
úroveň 5						480 856
úroveň 4					221 622	
úroveň 3				102 143		102 143
úroveň 2			47 077		47 077	
úroveň 1		21 697		21 697		21 697
úroveň 0	10 000		10 000		10 000	
úroveň -1		4 609		4 609		4 609
úroveň -2			2 124		2 124	
úroveň -3				979		979
úroveň -4					451	
úroveň -5						208

Obrázek 3.4: Předpokládaný vývoj hodnoty obětované části firmy (v tis. Kč). [6]

	perioda 0	perioda 1	perioda 2	perioda 3	perioda 4	perioda 5
úroveň 5						0
úroveň 4					0	
úroveň 3				0		0
úroveň 2			0		0	
úroveň 1		0		0		0
úroveň 0	0		0		0	
úroveň -1		3 391		3 391		3 391
úroveň -2			5 876		5 876	
úroveň -3				7 021		7 021
úroveň -4					7 549	
úroveň -5						7 792

Obrázek 3.5: Předpokládaný vývoj vnitřních hodnot prodejní opce (v tis. Kč). [6]

	perioda 0	perioda 1	perioda 2	perioda 3	perioda 4	perioda 5
úroveň 5						0
úroveň 4					0	
úroveň 3				0		0
úroveň 2			0		0	
úroveň 1		2 386		1 380		0
úroveň 0	**3 873**		3 290		2 164	
úroveň -1		4 853		4 453		3 391
úroveň -2			5 928		5 876	
úroveň -3				7 021		7 021
úroveň -4					7 549	
úroveň -5						7 792

Obrázek 3.6: Zpětný přepočet hodnoty americké prodejní opce (v tis. Kč). [6]

Na obrázcích 3.2 a 3.5 je vidět, že vývoj vnitřních hodnot je otočený. U kupní opce vnitřní hodnota s úrovněmi roste, u prodejní opce naopak klesá. Je tomu tak kvůli způsobu výpočtu této hodnoty ($S - X$ pro kupní opci, $X - S$ pro prodejní opci).

3.3.3 Další aplikace reálných opcí

V předchozí části bylo prokázáno, že standardní případy lze řešit pomocí standardních opčních metod. Reálné opce se využívají zejména v USA v ropném průmyslu, v těžebních odvětvích a v energetice. Lze se s nimi setkat také ve farmaceutickém a chemickém průmyslu, biotechnologiích, v letectví nebo ve zbrojním průmyslu. [13] V literatuře [6] jsou uvedené další příklady pro základní opce uvedené v klasifikaci 3.2. Tyto příklady se většinou řeší standardní konstrukcí binomického stromu, nebo pomocí Black-Scholesova vzorce. V některých situacích ale tyto přístupy použít nelze. Jedním takovým případem je výpočet spojený s Cloud computingem.

Jak už bylo jednou zmíněno, výhodou cloudu je mimo jiné flexibilita efektivně snižovat, nebo naopak zvyšovat výpočetní prostředky dle aktuální potřeby. Aby bylo míru flexibility možné vyjádřit, nelze použít pouze prodejní opci na snížení výkonu, nebo pouze kupní opci na jeho rozšíření, ale ideálně jejich kombinaci, která pokryje obě situace v jednom vypočteném stromě. Tomuto speciálnímu případu je věnovaná následující sekce.

3.4 Reálné opce v oblasti IT investic

V předchozích částech se čtenář seznámil s konkrétními výpočty hodnoty flexibility pomocí reálných opcí. Jak už ale bylo zmíněno, pro aplikaci metodiky na oblast IT investic je třeba vzorce upravit, což udělal vedoucí práce Ing. Náplava v článku, ve kterém se problematice IT investic věnoval [14]. Veškeré informace v této sekci s ním byly průběžně konzultovány a vychází mimo jiné také z draftu jeho disertační práce [15] zpracovávané pod vedením autora knihy *Reálné opce* [10] prof. Starým.

Mnoho začínajících podniků se jistě setkalo s otázkou, jestli pro svou IT infrastrukturu zvolit cloudovou platformu, nebo raději lokální (on-premise) řešení. Jak už bylo zmíněno, u cloudového řešení lze výpočetní prostředky efektivně přidávat i odebírat. U on-premise řešení tato flexibilita schází. Jakmile jednou firma zakoupí vybranou konfiguraci např. serveru, není možné v budoucnu výkon snížit a nechat si část investice vrátit. Navyšování výkonu je sice možné, ale v podstatě nevratné. Hodnotu, kterou podnik s využitím cloudu ušetří, je však velmi těžké odhadnout. Z tohoto důvodu představil Ing. Náplava model postavený na výpočtu hodnoty reálné opce pomocí binomického stromu, který tento odhad poskytuje.

V tomto modelu je přímo porovnávána flexibilita získaná využíváním cloudového modelu proti flexibilitě, kterou poskytuje tradiční hardware. Aby bylo možné docílit relevantního výsledku, je třeba zkombinovat několik opcí dohromady. Výsledkem modelu je pak částka, kterou by uživatel mohl teoreticky

ušetřit, pokud by dal přednost cloudu. Předpis pro první čtyři období je vidět na obrázku 3.9, který je umístěný na konci této kapitoly. Soubor ve formátu pro *Microsoft Excel* obsahující předpis pro 5 období je součástí přiloženého CD.

Pro porozumění zmíněného předpisu (3.9) je třeba nejprve definovat jednotlivé parametry a následně rozebrat jednotlivé situace, které mohou v průběhu výpočtu nastat.

Předně je třeba říci, že kalkulace vychází z obecných výpočtů hodnot reálných opcí. Z tohoto důvodu mezi parametry patří pravděpodobnosti p a d, které jsou čtenáři důvěrně známé z oblasti opcí (2.4). Jedná se totiž o pravděpodobnost růstu p a o doplňkovou pravděpodobnost poklesu $1 - p$ (zde označená symbolem d).

Symbol A_i pak označuje tzv. anuitu, tedy poplatek za on-premise prostředky v i-té úrovni. Anuita je pravidelná splátka, kterou je splácen dlouhodobý dluh. Zpravidla to bývá roční platba, v tomto případě je to platba za zakoupený HW a SW v dané úrovni každé periody binomického rozvoje vnitřních hodnot [16], kterou je třeba odvádět po celou dobu životnosti hardware. Předpokládá se přitom, že životnost hardware (a licencí) se rovná T, tedy času zbývajícího do expirace opce. Pokud je tedy on-premise výkon navyšován například ve třetí periodě, znamená to, že pro úplně korektní výpočet je třeba hned při navýšení připočítat tři diskontované anuity, které bude muset majitel doplatit po expiraci opce. Toto připočítání není povinné a je tedy čistě a jen na majiteli, zda si přeje životnost lokálního vybavení zahrnout do výpočtu a zvýhodnit tak cloud, u kterého tyto výdaje nejsou.

Pro stanovení výše částky, kterou je třeba při nákupu přičíst (A_{exp}), se využívá níže uvedeného vzorce. Celková částka je tvořená anuitami, které by bylo třeba zaplatit až po vypršení opce, tedy mimo sledované období. Protože jsou placeny dříve, než by správně měly být, je ve výpočtu zahrnutý diskont, který převádí budoucí hodnotu na současnou.

$$A_{exp} = \sum_{i=0}^{P-1} A \cdot e^{-r\frac{T\cdot(n-i)}{n}}$$

kde parametry tvoří:

- A_{exp} – částka reprezentující anuity přesahující životnost opce,
- P – aktuální perioda, ve které se provádí nákup,
- A – anuita pro úroveň, ve které se provádí nákup,
- e – základ přirozeného logaritmu,
- r – bezriziková úroková míra,
- T – životnost opce,
- n – počet období.

Dále je důležité zmínit, že např. anuita A_3 představuje pouze poplatek za navýšení výkonu z úrovně dva do úrovně tři. Pokud by tedy do již existujícího serveru bylo třeba při přechodu z druhé do třetí úrovně dokoupit novou paměť, jednalo by se o poplatek pouze za ní, ne za celý server.

Dále platí, že:

$$(A_i = 0) \Leftrightarrow i \in \langle -n, 0 \rangle$$

$$(A_i \geq 0) \Leftrightarrow i \in \langle 1, n \rangle$$

Tedy jinak řečeno platí, základní vybavení pro uspokojení nulté úrovně je již zakoupeno a není ho tak třeba splácet.

Posledním parametrem je C_i, který reprezentuje paušální poplatek za cloudové služby v úrovni i. Na rozdíl od anuitního poplatku se jedná o celou částku, která se odvádí poskytovateli cloudových služeb v dané úrovni.

Pro C_i platí, že:

$$(C_i \geq 0) \Leftrightarrow i \in \langle -n, n \rangle$$

Jinak řečeno je třeba odvádět určitou částku i v nulté a v záporných úrovních. Pro A_i i pro C_i dále platí, že se mapují na hodnoty jednotlivých úrovní získaných z binomického stromu reprezentujícího vývoj hodnoty podkladového aktiva. Z vývoje této hodnoty je třeba odhadnout počet uživatelů infrastruktury a z toho plynoucí nároky na ní. Vyčíslení těchto nároků ve formě poplatků za paušál a nákladů na vylepšení on-premise prostředků pak představuje právě A_i a C_i.

Pro porozumění modelu je také důležité vědět, že v každém prvku stromu je vždy právě tolik sčítanců, kolik do něj vede cest z kořenu stromu. Jeden sčítanec tedy odpovídá jedné cestě stromem. Od té se dále odvíjí také uspořádání koeficientů p a d, které určují pravděpodobnost, že daná cesta nastane. Např. pravděpodobnost $p \cdot p \cdot d$ tedy znamená, že došlo dvakrát po sobě k navyšování a jednou ke snižování výkonu.

Nyní lze přistoupit k rozebrání jednotlivých situací, které v průběhu výpočtu vnitřních hodnot nastávají. Různé způsoby výpočtu je třeba volit na základě směru cesty, aktuální úrovně a počtu zaplacených on-premise úrovní. Jednou zakoupený HW (respektive SW) už totiž nelze vrátit, je s ním tedy třeba počítat v průběhu celé cesty. Teoretický základ pro tyto výpočty poskytuje tzv. opce s pamětí. Jedná se o speciální druh opce, pro kterou platí, že je závislá výhradně na trase, která byla provedena během určitého časového období. Hodnota opce tedy závisí na každém kroku, který podkladové aktivum uskutečnilo během životnosti opce. [17]

Konkrétní případy, které je při výpočtu třeba rozlišovat ilustruje obr. 3.7. Obě křivky v obrázku představují vývoj investice. Popisky p a d odpovídají pravděpodobnostem růstu a poklesu. Čísla u křivek pak odpovídají těmto situacím:

1. *Růst s dokupováním* – Jde o situaci, kdy je třeba zvyšovat výkon. V případě cloudu to řeší navýšení tarifu, v případě on-premise investice do vylepšení stávající konfigurace. Hodnotu v tomto bodě pak tvoří rozdíl

mezi součtem odváděných anuit pro cestu a poplatkem za navýšení tarifu (počítáno jako rozdíl současné úrovně vůči předchozí úrovni). Pokud podnik díky cloudu nic neušetří, vnitřní hodnota je v daném bodě nulová.

2. *Růst bez dokupování nad úrovní 0* – V tomto případě je nutné zvýšit výkon v cloudu, ale on-premise aktuálním nárokům stačí, protože už byla investice provedena v minulosti. Hodnota se proto počítá jako rozdíl placených anuit pro cestu a poplatku za cloud ušetřenému proti nulté úrovni (počítáno jako rozdíl aktuální úrovně vůči nulté úrovni). Pokud jsou poplatky za cloud vyšší než součet anuit, vnitřní hodnota je rovná nule.

3. *Růst bez dokupování pod, nebo na úroveň 0* – Tato situace reprezentuje stav, kdy je zakoupená on-premise konfigurace vždy nevýhodná. Úspory pro danou cestu se proto počítají jako součet ušetřených nákladů za on-premise, pokud by je šlo snížit, a částku ušetřenou v cloudu. Ušetřené náklady za on-premise jsou zde reprezentovány součtem anuit za nevyužité prostředky. Částka ušetřená v cloudu je získána rozdílem poplatku v aktuální (nulté, nebo nižší) a nulté úrovni.

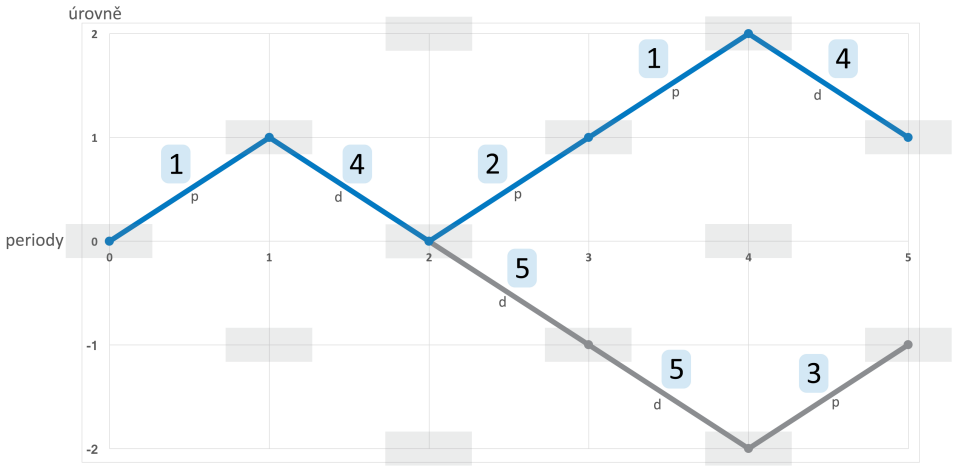
4. *Pokles nad, nebo na úroveň 0, zakoupeno více, než je aktuální úroveň* – Tato situace reprezentuje stav, kdy on-premise konfigurace není plně vytížená. V cloudu je tedy možné snížit tarif. Proto se vnitřní hodnota počítá jako součet nevyužitého on-premise výkonu a ušetřených poplatků za cloud díky snížení tarifu. Nevyužitý on-premise výkon představuje součet anuit za nevyužité prostředky. Úspora za snížení tarifu je počítána jako rozdíl poplatku za předchozí a aktuální úroveň.

5. *Pokles pod úrovní 0, zakoupeno více, než je aktuální úroveň* – Tento případ je podobný předchozímu. Jde o situaci, kdy je nutné snížit výkon až pod standardní konfiguraci. Proto se vnitřní hodnota počítá jako součet nevyužitého on-premise výkonu (součet všech zakoupených anuit) a ušetřené částky za cloud získané odečtením poplatku za aktuální úroveň od ceny za nultou úroveň.

Po vypočtení vývoje vnitřních hodnot pak už jen stačí provést standardní zpětný průchod stromem pro americkou opci. Výsledná hodnota investice je stejně jako v předchozích příkladech v kořenu stromu. Tato hodnota v daném kontextu představuje částku, kterou uživatel teoreticky může ušetřit v případě, že se rozhodne využívat cloudovou infrastrukturu. Čím vyšší tato hodnota bude, tím více peněz by šlo teoreticky ušetřit.

Výpočty v rámci této metodiky se provádí pomocí průchodů binomickým stromem. Tyto průchody svou implementací reprezentují zmiňovanou opci s pamětí.

V této části se ukázalo, že spočítat hodnotu investice do IT infrastruktury, je výrazně složitější, než spočítat běžnou opci. S každou další úrovní totiž počet členů a tedy i cest roste, což je pro manuální výpočet velmi náročné. Počet cest totiž odpovídá hodnotám v Pascalově trojúhelníku, takže se s každou



Obrázek 3.7: Možné situace při výpočtu hodnoty IT investice. [autor]

další úrovní jejich počet zdvojnásobuje (viz obrázek 3.8) a např. do dvacáté úrovně vede více než milion cest.

1											1									
2										1		1								
4									1		2		1							
8								1		3		3		1						
16							1		4		6		4		1					
32						1		5		10		10		5		1				
64					1		6		15		20		15		6		1			
128				1		7		21		35		35		21		7		1		
256			1		8		28		56		70		56		28		8		1	
512		1		9		36		84		126		126		84		36		9		1
1024	1		10		45		120		210		252		210		120		45		10		1

Obrázek 3.8: Pascalův trojúhelník určující počet cest do dané úrovně. [autor]

3.5 Nástroje pro výpočet hodnot reálných opcí

Nástroje používané pro podporu výpočtu hodnot reálných opcí lze rozdělit do třech kategorií.

První a přitom nejvíce rozšířenou skupinu tvoří tabulky ve formátu pro *Microsoft Excel*. Tyto tabulky často obsahují různá makra, která výpočet automatizují a vizualizují pomocí různých grafů. Použití Excelu je jednoduché, hodí se však primárně pro základní výpočty a malé vstupní instance (např. 10 period). Soubor umožňující počítat hodnoty základních opcí je součástí knihy Hodnota Flexibility [6]. Další soubor umožňující vypočítat prvních pět úrovní binomického stromu při hodnocení IT investice je součástí CD přiloženého u této práce. Proti oběma zmíněným souborům byly implementované postupy ověřovány.

Druhou skupinou jsou speciální matematické programy, jako je například Wolfram Mathematica [18] nebo Matlab [19]. Tyto nástroje jsou velmi efek-

tivní, mají široké spektrum využití a hodí se i pro velké instance. Jejich nevýhodou je ale vysoká cena a nutnost ovládat jazyk vybraného programu. Pro běžné uživatele se proto mohou jevit nedostupně.

Třetí skupinou jsou na míru psané programy, které jsou určené k výpočtu konkrétních příkladů. Takových programů není mnoho. Jedním příkladem je sada nástrojů od firmy Real Options Valuation Inc. [20]. Jedná se o placené nástroje určené k hodnocení investic, odhadování míry rizika a flexibility. Cena jedné licence se dle zvoleného programu pohybuje od 100 $ až do zhruba 9995 $, tedy v přepočtu od 3 do necelých 250 tisíc korun. Nejnovější nástroje ve zkušební verzi přitom působí zastarale a neintuitivně. Ani takto pokročilé a drahé nástroje ale neumožňují počítat speciální případy jako je zmíněný přístup k hodnocení IT investic. Jedná se totiž o novou, a tedy i obecně neznámou metodu. Absence dostupných nástrojů byla proto jedním z hlavních důvodů pro vypracování praktické části této diplomové práce.

	perioda 0	perioda 1	perioda 2	perioda 3	perioda 4
úroveň 4					$p*p*p*p*MAX(((A_1+A_2+A_3+A_4)-(C_4 - C_0)),0)$
úroveň 3				$p*p*p*MAX(((A_1+A_2+A_3)-(C_3 - C_0)),0)$	
úroveň 2			$p*p*MAX(((A_1+A_2)-(C_2 - C_0)),0)$		$p*p*d*p*MAX(((A_1+A_2)-(C_2 - C_0)),0) + p*d*p*p*MAX(((A_1+A_2)-(C_2 - C_0)),0) + d*p*p*p*MAX(((A_1+A_2)-(C_2 - C_0)),0) + p*p*p*d*(A_3+(C_3-C_2))$
úroveň 1		$p*MAX((A_1-(C_1 - C_0)),0)$		$p*p*d*(A_2 + (C_2-C_1)) + p*d*p*MAX((A_1-(C_1 - C_0)),0) + d*p*p*MAX((A_1-(C_1 - C_0)),0)$	
úroveň 0	0		$d*p*(0) + p*d*(A_1 + (C_1-C_0))$		$p*p*d*d*(A_1+A_2 + (C_1-C_0)) + p*d*p*d*(A_1 + (C_1-C_0)) + d*p*p*d*(A_1 + (C_1-C_0)) + d*d*p*p*(0) + p*d *d*p*(A_1+(0)) + d*p*d*p*(0)$
úroveň -1		$d*(C_0 - C_{-1})$		$p*d *d*(A_1+(C_0-C_{-1})) + d*p*d*(C_0-C_{-1}) + d*d*p*(C_0-C_{-1})$	
úroveň -2			$d*d*(C_0 - C_{-2})$		$p*d *d*d*(A_1+(C_0-C_{-2})) + d*p*d*d*(C_0-C_{-2}) + d*d*p*d*(C_0-C_{-2}) + d*d*d*p*(C_0-C_{-2})$
úroveň -3				$d*d*d*(C_0 - C_{-3})$	
úroveň -4					$d*d*d*d*(C_0 - C_{-4})$

nárůst potřeb na HW kapacity oproti předchozí nižší úrovni

pokles potřeb HW kapacity oproti předchozí vyšší úrovni

pravděpodobnost, kterou přenášíme z předchozí nižší úrovně a přidáváme pravděpodobnost nárůstu

pravděpodobnost, kterou přenášíme z předchozí vyšší úrovně a přidáváme pravděpodobnost poklesu

Obrázek 3.9: Předpis výpočtu hodnoty IT investice pro první čtyři periody [15]

Kapitola 4

Analýza a návrh řešení

Tato kapitola přímo navazuje na teoretickou část a poskytuje základy pro části následující. Hned v jejím úvodu jsou rozebrány požadavky definované v zadání doplněné o požadavky vedoucího práce. Po analýze pak už následuje popis návrhu implementovaného nástroje. V úvodu je popsaná základní datová struktura, za ní následuje doménový model a generování jednotlivých ukazatelů binomického modelu. Další části se věnují navrženému grafickému rozhraní nástroje a generování reportů. Závěr kapitoly je věnovaný zvolené technologii a návrhovému vzoru.

Protože je kód aplikace včetně uživatelského rozhraní v angličtině, při popisování jednotlivých struktur a částí systému se v této kapitole vyskytují anglické výrazy, které odpovídají reálně použitým názvům v kódu.

4.1 Cíle práce a využití

Hlavním cílem této práce bylo navrhnout a implementovat experimentální podpůrný nástroj, který jeho uživateli pomůže s výpočty hodnot reálných opcí a hodnot IT investic pomocí binomického modelu. Navržená aplikace je primárně určená pro využití v akademické sféře, při vhodné identifikaci vstupních dat ale najde své využití i v praxi. Jak bylo zmíněno v teoretické části, reálné opce mají široké využití a dostupných nástrojů pro práci s nimi není mnoho. Díky sdílení způsobu výpočtu s finančními opcemi lze navíc nástroj využívat také pro experimenty s nimi. Cílem přitom nebylo vytvořit aplikaci, která by vynikala například vzhledem, ale vytvořit funkční podpůrný nástroj, který svému uživateli ulehčí práci. Jedná se o program na míru, který dokáže nahradit aktuálně využívané nástroje (především interaktivní sešity pro *Microsoft Excel*). Předtím, než však bude možné přejít k jejímu návrhu, je třeba v zadání identifikovat nejdůležitější požadavky, čemuž je věnovaná následující sekce.

4.2 Analýza požadavků

Na navrhovaný nástroj bylo od začátku kladeno několik základních nároků. Tyto nároky lze rozdělit na funkcionální, tedy na požadavky na funkčnost

samotné aplikace, a nefunkcionální, které definovaly cílovou platformu, požadavky na výkon a design.

4.2.1 Požadavky na funkčnost

- *Binomický model* – Hlavním funkčním nárokem, který definuje celou aplikaci, je požadavek využívat pro výpočty hodnot opcí binomický model (2.4.1). Ten má jasně definované vzorce, pomocí kterých se konstruuje binomický strom (viz obrázek 2.3), s kterým se pak dále pracuje.

- *Výpočet hodnoty IT investice 3.4* – Krom obecných reálných opcí musí aplikace podporovat také výpočet rozvoje vnitřních hodnot a celkové hodnoty IT investice, který byl definovaný v sekci 3.4. Pro výpočet této metodiky je nutné implementovat zmiňovanou opci s pamětí.

- *Možnost vypnout, nebo zapnout dopočítávání anuit* – Tento požadavek souvisí s výpočtem hodnoty IT investice. Spočívá v možnosti určit, zda se při každém dokoupení on-premise techniky mají do úspor ve prospěch cloudu započítávat diskontované anuity, které je třeba doplatit po expiraci opce.

- *Skládání opčních stromů za sebe* – Požadavkem je možnost kombinovat binomické stromy skládáním za sebe. Stromy se musí chovat jako oddělené opce, které začínají z úrovně předchozího stromu. Nové stromy by měly být primárně generované z poslední úrovně (tedy z listů stromu). Generování stromů z vnitřních uzlů by mělo být kvůli možným nepřesnostem možné vypnout. Grafické prostředí musí být navržené tak, aby bylo možné postavit z poslední periody více stromů současně.

- *Změna parametrů, stromů a různé průchody* – Tyto požadavky jsou stavebními kameny pro další funkcionalitu. Jak napovídají vzorce, binomický strom je pro dosažení relevantních výsledků nutné procházet různými směry a přitom počítat jednotlivé ukazatele. Bez implementace různých průchodů stromem by ho totiž nebylo možné korektně generovat, zobrazovat, ani upravovat.

- *Ukládání do souboru* – Za nezbytnou součást aplikace je dále považována funkcionalita umožňující ukládání rozdělané práce do souboru s možností následného zpětného načtení.

- *Generování reportů* – Důležitou funkcionalitou je možnost generovat ze stromů vhodné reporty.

- *Nastavení počtu desetinných míst* – Jedním z nároků je možnost určit počet desetinných míst, na který se zaokrouhlí výstupy.

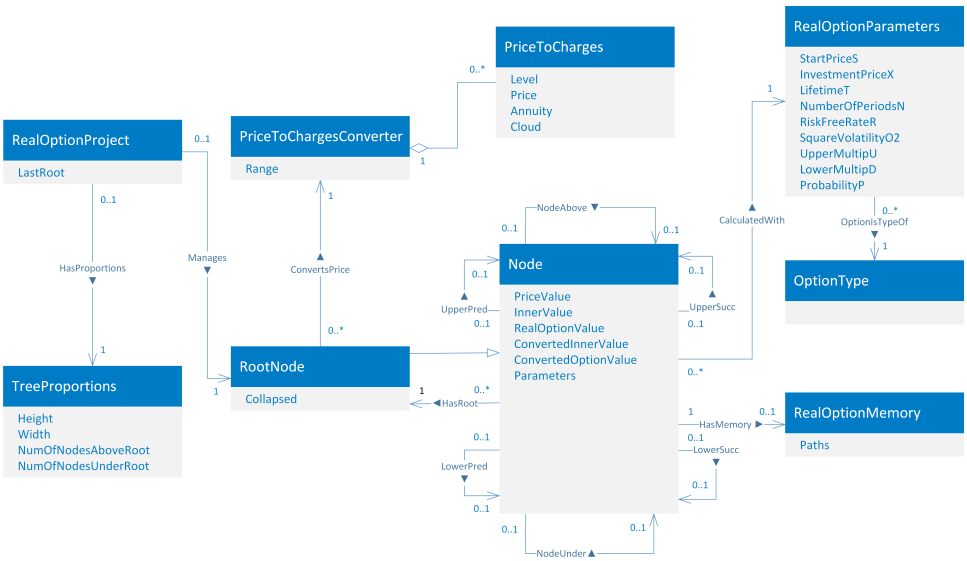
- *Tvoření snímků obrazovky* – Posledním funkčním požadavkem je možnost jednoduše uložit snímek obrazovky z vývoje binomického stromu.

4.2.2 Nefunkcionální požadavky

- *Desktopová aplikace* – Hlavním nefunkcionálním požadavkem je cílová architektura. Navržený nástroj má být implementovaný ve formě desktopové aplikace.
- *Microsoft Windows* – Požadovanou platformou, pod kterou bude aplikace spustitelná je Microsoft Windows na běžných osobních počítačích.
- *Jednoduchost nasazení* – Nástroj musí být jednoduše spustitelný, s instalací by si měl umět poradit i člověk, který v IT není zběhlý.
- *Neomezující rychlost výpočtu* – Rychlost výpočtu a odezvy nesmí být omezující, z druhé strany ale nejde o hodnotící faktory. Zásadní je především funkčnost.

4.3 Návrh

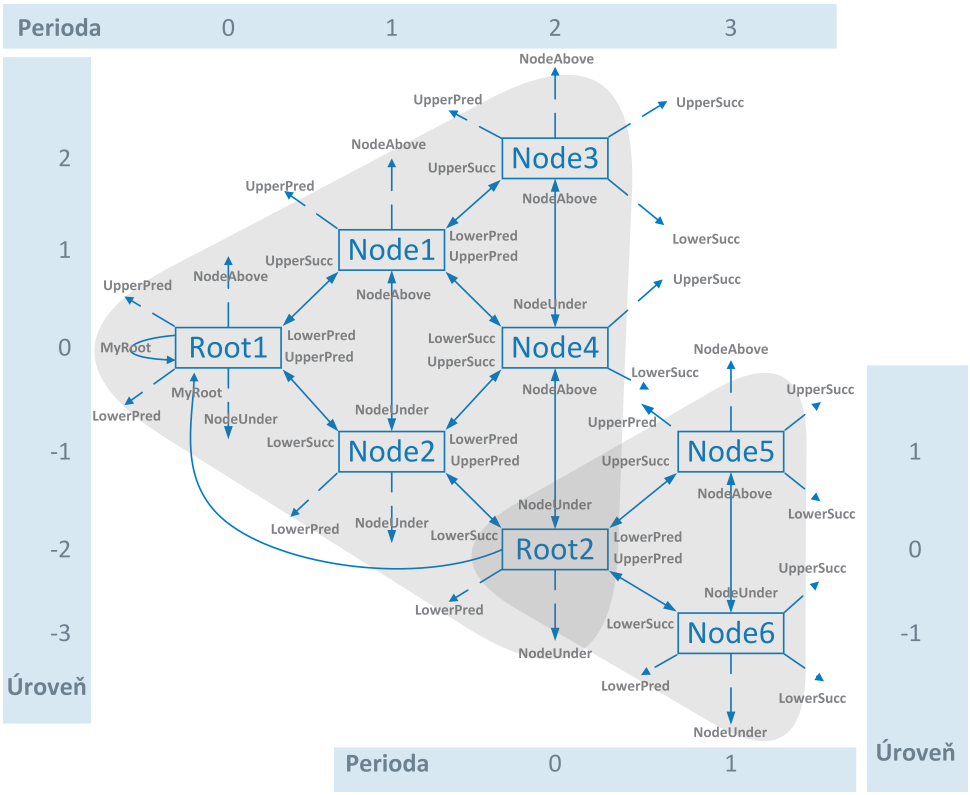
Na základě požadavků byl navržený doménový model, který reprezentuje jádro aplikace. To se stará o konstrukci stromu, výpočty jednotlivých ukazatelů a různé průchody. Tento model je vidět na obrázku 4.1. Hlavní třídou je *RealOptionProject*, což je třída reprezentující celý opční projekt. Třída si drží referenci na kořen hlavního stromu a na strukturu, která reprezentuje aktuální rozměry binomického stromu (*TreeProportions*). Samotný strom je reprezentovaný pomocí tříd *RootNode* a *Node*. Tato struktura je podrobně popsaná v následující sekci. Instance těchto tříd si dále drží reference na parametry nutné k výpočtu hodnot opce (*RealOptionParameters*), na opční paměť *RealOptionMemory* a na převodník aktuální ceny na poplatky za on-premise prostředky (anuity) a poplatky za cloudovou infrastrukturu.



Obrázek 4.1: Doménový model jádra aplikace. [autor]

4.3.1 Datová struktura pro binomický strom

Jak je vidět z doménového modelu, základním stavebním kamenem pro správnou funkčnost celého nástroje je datová struktura reprezentující binomický strom. Požadavky na strukturu se týkaly především možností procházet strom od kořene k listům i zpět nebo provádět průchod jednou celou periodou nahoru i dolů.



Obrázek 4.2: Navržená datová struktura reprezentující binomický model. [autor]

Obrázek 4.2 napovídá, že základní datovou strukturu netvoří strom, jak by mohl evokovat název modelu, ale orientovaný graf. Použití stromové struktury je nevhodné z důvodu potřeby zpětných referencí a kvůli vlastnosti binomického modelu, pro který platí, že spodní následník vrchního uzlu je zároveň vrchním následníkem spodního uzlu. Na obrázku 4.2 je tato vlastnost vidět např. mezi uzly 1, 2 a 4.

Dále platí, že v poslední periodě stromu mohou začínat další stromy. Jejich kořenový uzel je součástí předchozího stromu, žádné další reference na předchozí strom ale vnitřní uzly nemají, což je vidět například mezi uzly 4 a 5. Přesto, že by na sebe běžně měli vzájemně ukazovat, není tomu tak, protože mají různé kořeny.

Každý uzel obsahuje celkem šest referencí na uzly v jeho okolí plus jeden odkaz na svůj kořen (Root):

- *UpperSucc* – Zkráceno z anglického výrazu *Upper Successor*, tedy *vrchní následník*. Jedná se o odkaz mířící na vrchního následníka v grafu. Uzly

v poslední periodě tuto referenci nemají nastavenou, protože už žádné následníky nemají.

- *LowerSucc* – Tento název pochází z anglického *Lower Successor*, tedy *spodní následník*. Také pro tento odkaz platí, že ho uzly v poslední periodě nemají nastavený.

- *UpperPred* – Nezkráceně *Upper Predecessor*, neboli *vrchní předchůdce* odkazuje na uzel, který se nachází vlevo nahoře vůči aktuálnímu uzlu. Tuto referenci nemá nastavenou hlavní kořen a všechny uzly na diagonále vedoucí z kořene do pravého horního uzlu.

- *LowerPred* – Původně *Lower Predecessor* reprezentuje odkaz na *spodního předchůdce* v grafu. Jedná se o zpětnou referenci, kterou nemá nastavený hlavní kořen a všechny uzly na diagonále vedoucí z kořene do pravého dolního uzlu.

- *NodeAbove* – V češtině *uzel nad*, tedy odkaz na uzel, který se nachází nad jeho majitelem. Tato reference je vhodná zejména při procházení jedné periody, ale využívá se i při generování stromu. Díky zpětným referencím (*LowerPred* a *UpperPred*) není sice odkaz nezbytně nutný, práce se stromem je ale díky němu pohodlnější. Není nastavený pro všechny uzly na diagonále vedoucí z kořene do vrchního pravého uzlu.

- *NodeUnder* – Česky *uzel pod* představuje odkaz na uzel pod jeho majitelem, jedná se o obrácenou referenci k *NodeAbove*. Své využití nachází hlavně při průchodech vybranou periodou. Pro uzly na diagonále vedoucí z kořene do pravého dolního uzlu odkaz není nastavený.

- *MyRoot* – V překladu *můj kořen* reprezentuje zpětnou referenci na předchozí (můj) kořen ve stromě. Pro běžné uzly ($Node_i$) odkazuje na kořen aktuálního stromu. U kořenů odkazuje na předchozí kořen ve stromě. Využívá se zejména ve složitějších instancích, kdy je za sebou poskládáno více opcí. Pro hlavní kořen odkaz ukazuje sám na sebe.

Datová struktura reprezentující binomický strom je stejná pro všechny části mezivýpočtu. Díky tomu není nutné držet v paměti několik instancí stromů pro každý z pohledů na něj (vývoj spotové ceny, vývoj vnitřních hodnot atd.), ale stačí pouze jedna instance struktury, která obsahuje všechny potřebné hodnoty. Konkrétně se jedná o hodnotu reprezentující vývoj ceny podkladového aktiva (*PriceValue*), vnitřní hodnotu uzlu (*InnerValue*), hodnotu reálné opce (*RealOptionValue*), vnitřní hodnotu IT investice (*ConvertedInnerValue*) a hodnotu opce IT investice (*ConvertedOptionValue*).

Hodnoty *PriceValue* a *InnerValue* je přitom možné určit už při vytváření uzlů, pro určení hodnoty *ConvertedInnerValue* je třeba nejprve znát konverzní tabulku sestavenou na základě vývoje ceny podkladového aktiva. Až na závěr lze zpětným průchodem stromem spočítat hodnoty *RealOptionValue* a *ConvertedOptionValue*.

Kořenový uzel rozšiřuje standardní uzel a přidává mu krom rozlišení typu navíc informaci o tom, zda je aktuální strom sbalený (*Collapsed*) a tedy je vidět pouze kořen, nebo zda je rozbalený.

4.3.2 Vytvoření stromu

Aby bylo možné přejít k dalším výpočtům, je nutné nejprve vygenerovat výše popsanou strukturu. K tomu je zapotřebí mít všechny parametry potřebné pro výpočet základní opce. Jedná se konkrétně o aktuální cenu podkladového aktiva (S), realizační cenu (X), dobu životnosti opce (T), míru volatility (σ^2), bezrizikovou úrokovou míru (r), počet období (n) a typ opce (*Call/Put*). Z těchto parametrů se spočítají hodnoty u a d, pomocí kterých se pak generuje vývoj spotové ceny a vývoj vnitřních hodnot.

Samotné vytváření pak probíhá postupně od kořene, přičemž aktuální uzel vždy vytváří své následníky. Spotová cena vrchního následníka (*UpperSucc*) se generuje jako $PriceValue * u$, spotová cena spodního následníka (*LowerSucc*) jako $PriceValue * d$. Následně je třeba rekurzivně udělat to samé také pro následníky. Je přitom nutné hlídat vlastnost, která je vidět mezi uzly 1, 2 a 4 na obrázku 4.2, tedy na to, že spodní následník vrchního uzlu je zároveň vrchní následník spodního uzlu. V praxi to znamená, že je třeba před vytvořením následníka nejprve zkontrolovat následníky okolních uzlů pro případ, že už by byl vytvořený. Ukončovací podmínka rekurze je dosažení požadovaného počtu období (n).

Po vytvoření stromu je možné přejít k výpočtu hodnoty IT investice a hodnoty reálné opce.

4.3.3 Hodnota IT investice

Pro vypočtení hodnoty IT investice bylo dále třeba navrhnout vhodnou strukturu pro implementaci opce s pamětí a vhodný algoritmus, který pokryje všechny případy, které mohou při výpočtu nastat (viz 3.4). K těmto účelům slouží třída *RealOptionMemory*, na kterou si drží referenci každý uzel. Součástí každé instance *RealOptionMemory* je pak atribut *Paths*, který reprezentuje jednotlivé cesty v paměti.

Jak ilustruje obrázek 4.3, strom je po zadání hodnot konverzní tabulky, která převádí aktuální cenu podkladového aktiva na poplatky za cloudové služby a on-premise prostředky, třeba projít a pro každou z cest spočítat vnitřní hodnotu IT investice.

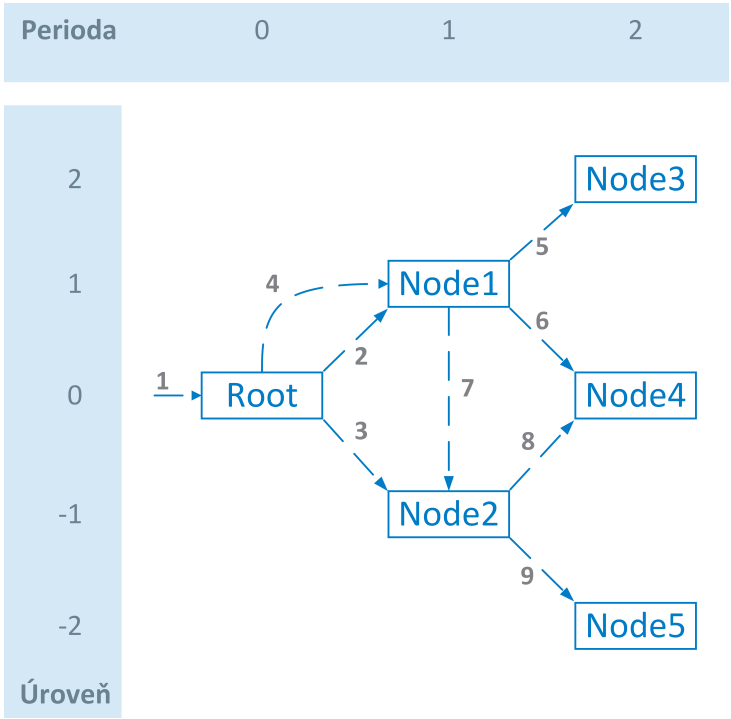
Aby nebylo nutné procházet celý strom vždy od začátku je v každém uzlu právě zmíněná *RealOptionMemory*. Ta umožňuje předchozí cesty cachovat (ukládat do mezipaměti), díky čemuž pak stačí pro získání hodnoty uzlu projít pouze cesty přímých předchůdců. Ty se pak uloží do paměti následníka, čímž se postupně z jednoduchých cest postupně stávají složené dlouhé cesty vedoucí až do poslední periody.

Z analýzy metodiky bylo zjištěno, že cestu lze reprezentovat pouze jednou proměnnou, která reprezentuje počet zaplacených úrovní z hlediska on-premise prostředků. Na první pohled by se mohlo zdát, že je nutné jí reprezentovat

ještě minimálně pravděpodobností, ale není tomu tak. Pravděpodobnost totiž lze odvodit z aktuální úrovně a periody. Její hodnota je totiž i přes různé pořadí členů díky násobení pro všechny cesty vedoucí do uzlu stejná. Při průchodu si pak stačí hlídat aktuální úroveň a periodu. Díky převodní tabulce a prolinkování grafové struktury tak, jak bylo popisováno, lze z těchto údajů odvodit patřičný vzorec a dosadit do něj.

Celý výpočet vnitřních hodnot IT investice probíhá tak, že je nejprve nastavena počáteční hodnota (krok 1) a poté jsou do vnitřních hodnot obou následníků (kroky 2 a 3 nebo třeba 5 a 6) přičteny hodnoty ze vzorce zvoleného na základě aktuální úrovně, periody, směru cesty a počtu zaplacených anuit pro danou cestu. Takto je třeba projít všechny cesty a pro každou z nich přičíst do následníka s ní související člen a zároveň mu danou cestu přidat do jeho paměti. V případě, že je nutné pro danou cestu přikoupit on-premise prostředky a zároveň je zapnuté dopočítávání anuit, je dále nutné ke členu reprezentující cestu přičíst diskontovaný doplatek reprezentující životnost prostředků po konci sledovaného období.

Díky tomu, že se vnitřní hodnota počítá vždy pro následující periodu, lze při opouštění uzlu jeho paměť vymazat a přesunout se do uzlu pod ním. Po průchodu celé periody se vždy výpočet přesouvá do vrchního uzlu následující periody (krok 4), která se opět prochází odshora dolů (krok 7). K průchodu periody se využívá referencí *NodeUnder*.



Obrázek 4.3: Průchod stromem při výpočtu vnitřní hodnoty IT investice. [autor]

4.3.4 Zpětný průchod stromem

Finální hodnota reálné opce i hodnota opce IT investice se počítá při zpětném průchodu stromem. Protože se finální struktura může skládat z více opcí a tedy i z více stromů, bylo nutné navrhnout algoritmus tak, aby byla vždy hodnota propočítána od naposled vytvořeného stromu postupně přes všechny jeho rodičovské stromy až k úplnému kořenu, kde je uložený celkový výsledek. K zajištění toho, aby byly přepočítány opravdu všechny podstromy v cestě, lze využít reference *MyRoot* v kořenových uzlech. Díky ní je možné výpočet soustředit pouze do jednoho podstromu, což ho značně zjednoduší. Jediné, co je pak třeba zajistit, je provedení průchodu pro všechny kořenové uzlu v cestě.

V praxi tedy stačí projít strom po diagonále z kořene až do nejvyššího prvku poslední periody a následně procházet periodu odshora dolů a počítat při tom požadované hodnoty. K výpočtu se využívá mimo jiné také zpětných referencí *UpperPred* a *LowerPred*, protože je nutné neustále porovnávat data z aktuální periody vůči periodě předchozí (viz vzorce v sekci 2.4).

Při jednom průchodu se pak počítá hodnota reálné opce i opce IT investice. Jediný rozdíl je, že se pro každý z vývojů využívá jiných hodnot.

4.3.5 Řetězení opcí

Dalším požadavkem, který bylo třeba splnit bylo skládání opcí za sebe. Jako reakce na tento požadavek byly navrženy obě varianty přístupu k tomu, odkud lze stromy vytvářet. První je standardní přístup, který umožňuje uživateli přidávat nové opční stromy vždy od poslední periody již vytvořeného stromu a druhý přístup umožňující řetězit nové opce i za vnitřní uzly.

V prvním přístupu je třeba vzít uzel, od kterého začíná nový strom a vytvořit z něj kořenový uzel (*RootNode*). Přitom je třeba správně přeřadit všechny reference z okolních uzlů a následně z něj spustit generování nového podstromu.

V druhém přístupu, který je dle požadavků standardně vypnutý pro možné nepřesnosti, lze vytvářet nové opce z vnitřních uzlů. V takovém případě je třeba odříznout následující periody a vytvořit tak z periody vybraného uzlu poslední periodu. Možné nepřesnosti mohou vznikat právě kvůli tomuto zkrácení, které ovlivní parametry použité pro konstrukci původního stromu.

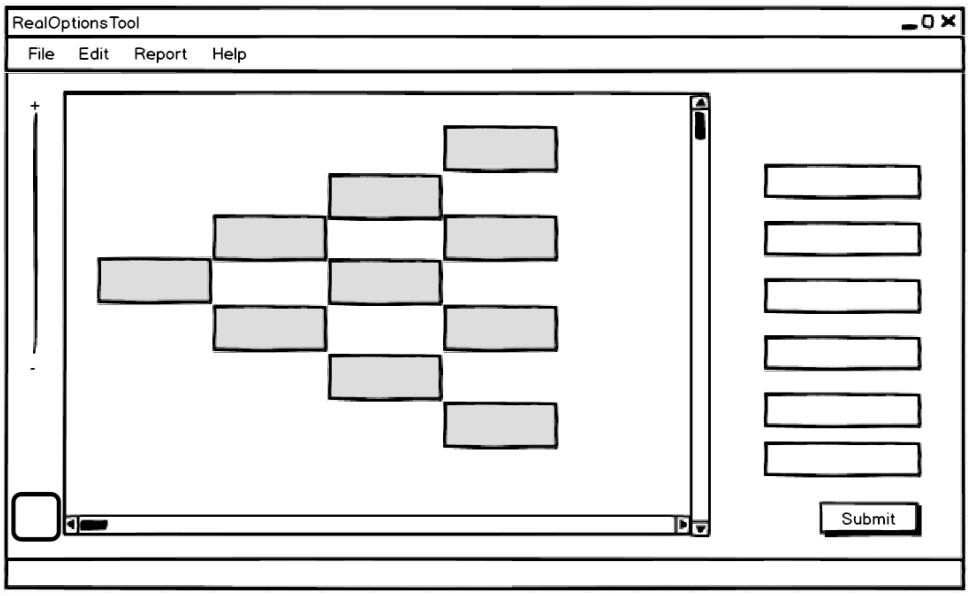
4.3.6 Grafické rozhraní

Po navržení způsobů výpočtu jednotlivých ukazatelů bylo třeba rozhodnout, jakým způsobem budou výsledné struktury prezentovány uživateli. Nejdůležitější částí grafického uživatelského rozhraní (GUI) je beze sporu plocha, na které je zobrazený vývoj binomického stromu. Inspirace pro tuto část tvořily obrázky stromů v literatuře [6] a nástroje pro MS Excel. Bylo rozhodnuto, že se výsledek bude zobrazovat do pomyslné tabulky. Plocha pro vykreslování je při tomto přístupu rozdělena na stejně velká pole, do kterých jsou tisknuty jednotlivé uzly podobně, jako by šlo o tabulku. Tento přístup je výhodný pro

svou jednoduchost, přehlednost a zároveň univerzálnost použití. Lze jej totiž použít nejen pro vykreslování v aplikaci, ale také pro exportování požadovaných reportů do tabulky pro MS Excel. Jeho nevýhodou je zobrazování do 2D prostoru. Pro splnění požadavku týkající se vytváření více stromů z poslední periody současně je tedy třeba implementovat skrývání jinak překrývajících se částí grafu. Možnost skrýt, nebo naopak rozšířit strom by měla být na kořenovém uzlu na první pohled viditelná.

Další neméně důležitou částí je formulář umožňující zadávat a měnit parametry mezivýpočtu. V GUI měla být také možnost uložit snímek obrazovky, vygenerovat požadovaný report a několik dalších funkčních prvků, které lze umístit do menu.

Návrh, ze kterého vycházela implementace, je (po digitalizaci) vidět na obrázku 4.4. Na standardním místě, v liště nahoře, je umístěné menu obsahující tlačítka pro generování reportů, ukládání práce a dalších požadovaných funkcionalit nástroje. Pod ním pak leží zbytek prvků umístěných do mřížky. Vlevo nahoře je umístěný prvek pro zvětšování a zmenšování pohledu na strom. Pod ním je tlačítko pro uložení snímku stromu. Vpravo od nich už se pak nachází plocha pro zobrazování stromu, která celému GUI dominuje. Vpravo od ní se pak nachází formulář pro vkládání a úpravy parametrů mezivýpočtu.



Obrázek 4.4: Návrh grafického rozhraní. [autor]

4.3.7 Generování reportů

Jako formát pro generování vhodných reportů byl po dohodě s vedoucím práce zvolený xlsx pro *Microsoft Excel*. Tabulka pro Excel totiž umožňuje snadno zobrazit strom, jak je popsané výše v sekci 4.3.6. Není v ní sice možné snadno implementovat zmiňované skrývání překrývajících se uzlů, ale lze v ní snadno přidávat další listy a zobrazit tak všechny možné pohledy na strom

i všechny jeho podstromy. Pro zobrazení více informací o daném uzlu lze využít komentářů k jednotlivým buňkám.

Průchod stromem při generování exportu se tedy od ostatních průchodů liší. Bylo třeba ho navrhnout tak, aby se při něm vytisklo maximum podstromů, které se nepřekrývají, ale zároveň tak, aby se data na jednotlivých listech zbytečně neopakovala. Pokud datová struktura obsahuje překrývající se stromy, jednoduše se v první iteraci překrývající se strom přeskočí a vytiskne se v následující iteraci do dalšího listu. K tomuto postupu je mimo jiné třeba využít pomocnou proměnnou, ve které je uložená informace o tom, který kořen (a tedy i strom) už byl vytisknutý a který ještě ne. Na základě této informace mohou být při průchodu stromem následníci aktuálního kořene buď tištěni dál, nebo přeskočeni.

Protože může každého uživatele zajímat trochu jiný pohled na strom, musí být v aplikaci implementovaná možnost zvolit si, jaký průběh má být součástí reportu. Z pohledu návrhu se nejedná o žádný problém, půjde pouze o to si zvolit, která hodnota má být při průchodu stromem tisknuta do reportu.

4.3.8 Nastavení

Součástí funkčních požadavků je také několik nároků na konfiguraci aplikace. Konkrétně se jedná o možnost vypnout, nebo zapnout dopočítávání anuit, dále jde o volbu počtu desetinných míst a na závěr možnost vypnout, nebo zapnout tvorbu nových opčních stromů z vnitřních uzlů již vytvořené struktury. Pro pokrytí těchto požadavků je třeba implementovat okno, kde bude tato konfigurace umístěna. Nastavení by mělo zůstat stejné i po restartu aplikace.

4.4 Zvolené technologie a návrhový vzor

Pro implementaci popisovaného nástroje byl na základě analýzy požadavků zvolený framework Microsoft .NET, jazyk C# a technologie WPF.

4.4.1 Microsoft .NET Framework

Požadovanou cílovou platformou byl operační systém Microsoft Windows. Z tohoto důvodu byl pro implementaci aplikace zvolen standardní framework využívaný k vývoji pro tuto platformu a tím byl Microsoft .NET Framework. Jedná se o technologii, která se využívá k vývoji desktopových i webových aplikací různého rozsahu. Není přímo závislý na jazyku, nicméně nejčastěji využívané jazyky jsou C# (C#.NET), Visual Basic (VB.NET) a C++ (C++.NET). Skládá se z *Common Language Runtime* (CLR) a *.NET Framework Class Library*.

CLR tvoří základy .NET, lze si jej představit jako agenta, který spravuje kód v době provádění. Dále poskytuje základní služby jak je správa paměti, vláken a zároveň prosazuje přísné zásady bezpečnosti a současně prosazuje přísnou bezpečnost typu a jiné formy správnosti kódů, které podporují bezpečnost a robustnost. CLR má na starost především běh a kompilaci aplikací. [21]

Druhou neméně důležitou součástí frameworku je knihovna tříd .NET. Jedná se o sadu opakovaně použitelných typů, které jsou úzce propojeny s běhovým prostředím CLR. Knihovna je objektově orientovaná a poskytuje rozšiřitelnou funkcionalitu. Díky ní framework umožňuje snadno provádět celou řadu běžných programovacích úloh.

4.4.2 Jazyk C#

Jak již bylo řečeno, pro .NET lze vyvíjet v mnoha objektových jazycích. Nejčastěji využívaným jazykem je C#, který byl zvolený také pro implementaci cílového nástroje.

Hlavní předností, pro kterou jsem si ho vybral, je především jeho provázanost s frameworkem .NET. C# je totiž stejně jako .NET vyvíjený firmou Microsoft. Za plus považuji také jeho *C-like* syntaxi, objektový přístup a práci s ukazateli.

Standard ECMA jazyk uvádí jako jednoduchý, moderní, univerzální objektově-orientovaný programovací jazyk. Pro jeho vývoj se nejčastěji používá Microsoft Visual Studio, ve kterém byla vyvíjena také navržená aplikace. [22]

4.4.3 Windows Presentation Foundation

Pro implementaci grafického prostředí byl vybrán *Windows Presentation Foundation* neboli WPF. WPF je framework pro komplexní tvorbu formulářových desktopových aplikací, který je součástí .NET od verze 3.0. Jedná se o nástupce technologie *Windows Forms* (WinForms), kterou jsem díky zkušenostem pro implementací také zvažoval. Nakonec jsem se ale rozhodl pro WPF. Dal jsem mu přednost především z toho důvodu, že se jedná o novější, preferovanou technologie, která v současné době nabízí více možností, než starší WinForms. Pro popis grafického rozhraní se u WPF používá značkovací jazyk XAML, který pomocí *Data Bindingu* umožňuje striktně oddělit vzhled od funkčnosti aplikace. *Data Binding* je technologie, pomocí které lze prvky grafického rozhraní oboustranně svázat s proměnnými. I když to není pravidlem, nejčastěji se WPF využívá ve spojení s návrhovým vzorem MVVM, pro který je WPF navržené a dle kterého byla navržena také tato aplikace. [23]

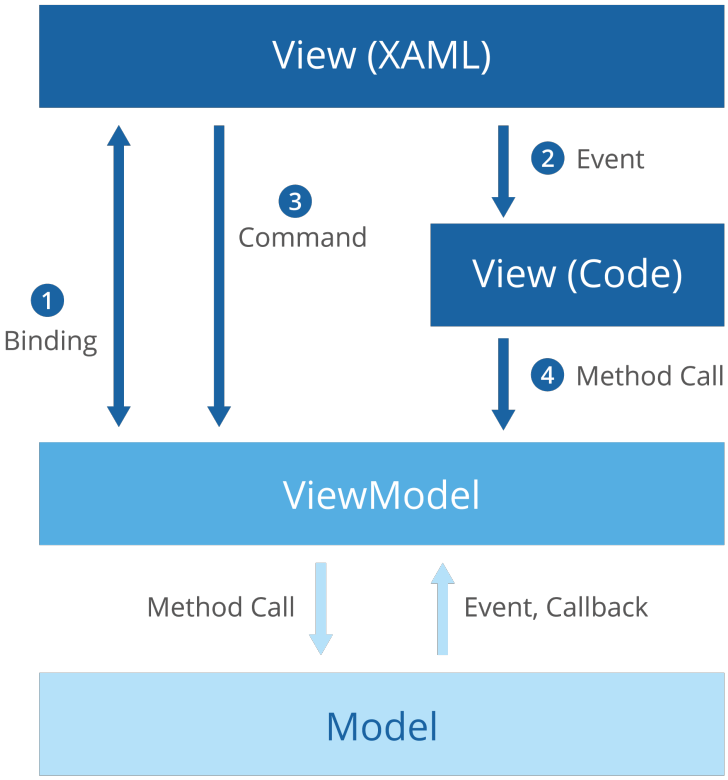
4.4.4 Návrhový vzor MVVM

Model-View-ViewModel (MVVM) je návrhový vzor, který je doporučovaný pro vývoj WPF aplikací. Jeho cílem je oddělit logiku aplikace od uživatelského rozhraní . Díky tomu se pak kód tolik neopakuje, vzniká ho méně, je přehlednější a snáz udržovatelný. Jak už napovídá název, princip vzoru spočívá v rozdělení aplikace do těchto tří vrstev [24][25]:

- *Model (M)* – Model představuje implementaci doménového modelu aplikace, který obsahuje datový model spolu s aplikační a validační logikou.

Model by neměl vědět o stavu ovládacích prvků a nikdy by neměl komunikovat s vrstvou View přímo, ale pouze prostřednictvím ViewModelu. Model ho notifikuje o změnách pomocí událostí, které ViewModel sleduje.

- *View (V)* – View je zodpovědné za definování struktury, rozložení a vzhledu, který vidí uživatel. V ideálním případě je View definováno čistým XAML jen s minimem kódu na pozadí obsahující business logiku. Pro svázání View s Modelem se využívá ViewModel, který se ve View nastavuje jako *DataContext*. Jeho vlastnosti jsou pak vázané na View. Komunikaci směrem k ViewModelu zajišťují příkazy (*Command*), informace o nutnosti překreslit View zajišťují události vyvolané ViewModelem.
- *ViewModel (VM)* – ViewModel funguje jako prostředník mezi View a Modelem. Jeho účelem je především udržování stavu aplikace a zpracování logiky zobrazování. ViewModel typicky interaguje s modelem pomocí volání jeho metod, Model pak informuje ViewModel o případných změnách pomocí vyvoláním událostí. Data, která ViewModel od Modelu získá, upraví do snadno zobrazitelné formy a zpřístupní je View, které má na starost jejich zobrazení uživateli.



Obrázek 4.5: Model-View-ViewModel, vytvořeno na základě [25].

Jednotlivé vrstvy a komunikaci mezi nimi ilustruje obrázek 4.5. Po přiblížení jednotlivých kroků návrhu je možné se přesunout ke konkrétní implementaci ve zvolené platformě, které je věnována následující kapitola.

Kapitola 5

Implementace a nasazení

Po seznámení se s jednotlivými kroky návrhu se čtenář může přesunout ke kapitole věnující se konkrétní implementaci popisovaného nástroje. Hned v úvodu kapitoly je popsaná struktura řešení rozdělená do sekcí inspirovaných použitým návrhovým vzorem MVVM. Za ní následuje obsáhlá sekce, která shrnuje průběh implementace, popisuje použité prvky a přibližuje problémy, které bylo při práci na projektu nutné řešit. Závěr kapitoly je pak věnovaný nasazení nástroje a zhodnocení implementační fáze.

5.1 Struktura řešení

V této sekci je popsaná struktura řešení. Vyskytují se zde anglické výrazy, které přesně odpovídají názvům jednotlivých komponent v implementované aplikaci. Struktura se skládá z několika hlavních oddílů a několika vedlejších pomocných složek. Hlavní složky tvoří:

- *Model* – V tomto balíčku je implementovaný doménový model (obrázek 4.1). Složka obsahuje veškerou business logiku, která zajišťuje jednotlivé výpočty.
- *View* – Jak název napovídá, tato složka obsahuje třídy a XAML soubory spojené grafickým rozhraním.
- *ViewModel* – Tato složka obsahuje jednotlivé ViewModel třídy, které mají na starost zajistit komunikaci mezi View a Modelem.
- *Reporting* – V této složce je uložená logika zajišťující exportování dat ve formě reportů určených pro *Microsoft Excel*.

Vedlejší (pomocné) balíčky jsou pak složeny z:

- *Commands* – Zde jsou uložené třídy implementující rozhraní ICommand určené ke komunikaci mezi View a ViewModelem.
- *Converters* – Tento balíček obsahuje pomocné třídy, které převádí vlastnosti ViewModelu na konkrétní atributy ovládacích prvků View.

- *Extensions* – Tato složka je složená z tříd, které rozšiřují funkcionalitu základních .NET tříd.

Výše uvedené složky pak doplňuje jedna statická třída *Utility*, která obsahuje pomocné metody.

5.2 Průběh implementace

Následuje průběh implementace. Sekce je členěna chronologicky dle toho, jak probíhala práce na projektu. Nejprve byly implementovány jednotlivé algoritmy a struktury na Business vrstvě (Model) spolu se základním grafickým rozhraním, které umožňovalo kontrolovat správnost implementace. Následně bylo grafické rozhraní dokončeno a práce pokračovala generováním reportů. Po dokončení reportů bylo implementováno ukládání rozdělané práce do souboru a na závěr byly složitější výpočty přesunuty do paralelních vláken běžících na pozadí.

5.2.1 Sestavení stromu

Postup generování stromu byl popsaný v kapitole věnující se návrhu (4.3.2). Implementace navrženého postupu probíhala bez větších problémů. Implementovaný algoritmus strom vytváří postupně s využitím rekurze. V průběhu vytváření vždy kontroluje následníky okolních uzlů, v případě, že potřebný následník již existuje, pouze se nastaví reference na již vytvořenou instanci. V opačném případě je vytvořená nová instance. Hodnota spotové ceny pro daný uzel je předávána v konstruktoru spolu s referencí na parametry použité k vytvoření stromu a odkazem na kořenový uzel. Vnitřní hodnota uzlu je počítána dynamicky při volání *Property* pro získání její hodnoty. Po vytvoření, nebo získání reference následníka se nastaví členy *NodeAbove* a *NodeUnder*, díky kterým v každé periodě uzly tvoří oboustranně zřetězený spojový seznam. Pak už stačí jen rekurzivně zavolat metodu a generovat další prvky až do splnění požadovaného počtu period.

Vzhledem k nutnosti mít pro výpočet velké množství parametrů, byly v průběhu práce využívány na pevno zadané parametry z literatury, na kterých byla ověřována správnost výpočtu. Jako velmi důležité se ukázalo kontrolovat, aby byly v rekurzi nové uzly vytvářeny opravdu jen jednou. Při vynechání jedné kontrolní podmínky se výpočet zpomalil z řádu několika milisekund na jednotky desítek sekund pro vstupní instanci o třiceti periodách, chyba však byla upravena hned na počátku implementační fáze.

5.2.2 Spočítání hodnoty IT investice

Algoritmus pro spočtení hodnoty IT investice byl taktéž popsaný v kapitole o návrhu (4.3.3). Pro správné pokrytí veškerých případů, které byly popsány v metodice bylo nutné naprogramovat popisovaný průchod stromem spolu s metodou počítající hodnotu pravděpodobnosti pro daný uzel. Dále bylo

třeba si při průchodu držet informace o aktuální úrovni, periodě a o všech cestách vstupujících do uzlu.

V úplně první verzi implementace bylo k řešení pro jednoduchost přistupováno naivně. Jednotlivé cesty (atribut *Paths*) v paměti reálné opce byly při naivním přístupu reprezentovány standardní kolekcí, která obsahovala celočíselné proměnné reprezentující počet anuit, které je třeba odvádět s každým dalším obdobím. Tento přístup fungoval, ale vzhledem k vlastnosti metodiky, která kopíruje počtem cest hodnoty Pascalova trojúhelníku (3.8), byl výpočet pro větší počet úrovní velmi pomalý a paměťově náročný. Bylo tedy třeba strukturu zoptimalizovat. Klíčem úspěchu přitom bylo si uvědomit, že je velmi pravděpodobné, že se budou cesty díky jednoduché reprezentaci často opakovat.

Kvůli opakování cest byla kolekce nahrazena datovou strukturou *Dictionary*, která je v .NET implementována hashovací tabulkou složenou z dvojic klíčů a hodnot. Jako klíč byla zvolena hodnota reprezentující počet placených anuit (tedy nejvyšší dosaženou úroveň) a jako hodnota byl zvolen počet takových cest. Toto vylepšení algoritmus diametrálně vylepšilo, a to jak z pohledu rychlosti, tak z pohledu paměti. Výsledky přitom zůstaly stejné.

Rozdíl byl pochopitelně obrovský, pro srovnání výpočet vstupních dat o 25 periodách ($n = 25$) pomocí naivního řešení trval necelých 30 sekund s nárůstem využití paměti okolo 900 MB. Výpočet stejné vstupní instance s využitím optimalizovaného řešení trvá v řádu jednotek milisekund a paměťově je jeho průběh v rámci debuggeru Visual Studia stěží rozpoznatelný, zaznamenaný nárůst byl v jednotkách MB.

Výpočet vnitřních hodnot IT investice se provádí vždy po úpravě, nebo prvním vyplnění konverzní tabulky mezi náklady na cloudové a on-premise řešení.

5.2.3 Spočítání hodnoty opce

Zpětný průchod stromem a s ním související výpočet byl implementovaný formou několika rekurzivních metod ve třídě *Node* dle postupu popsaného v sekci 4.3.4. Pro správný propočet bylo třeba implementovat průchod stromem až do poslední periody, postupný průchod periodou a na závěr metodu, která zajistí, že bude budou průchody provedeny nad všemi stromy v cestě k hlavnímu kořenu.

Na výpočet hodnoty reálné opce i hodnoty IT investice stačí díky stejnému principu provést jen jeden průchod. Způsob výpočtu vychází z definovaných vzorců (2.4 a 3.4). Protože je zpětný průchod nutné vždy provádět od posledního postaveného stromu, je ve třídě *RealOptionProject* uložená reference na nejnovější kořenový uzel, pro který jsou patřičné metody volané jako první. Následně je to samé provedeno také pro kořeny v cestě (přes reference *MyRoot*).

Zpětný propočet se provádí vždy po spočítání vnitřních hodnot IT investice.

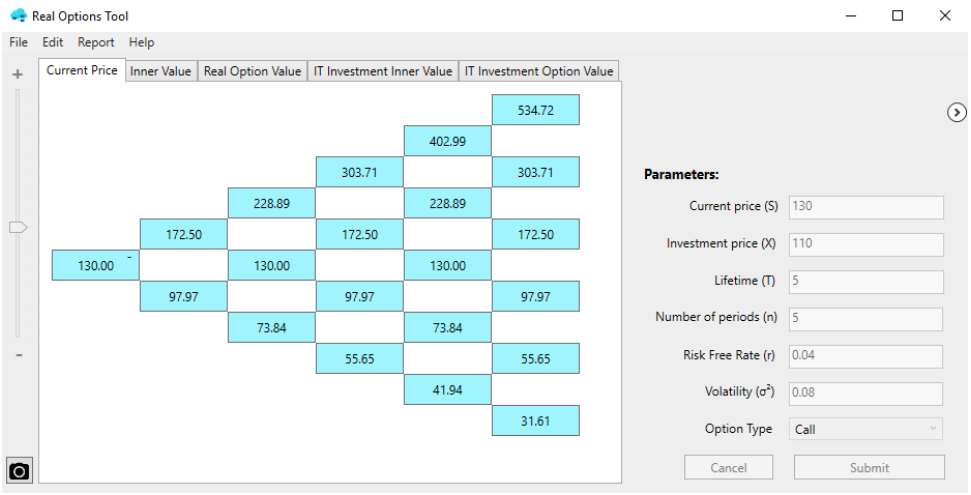
5.2.4 Řetězení opcí

Na základě návrhu (4.3.5) byly implementovány dva přístupy k řetězení opcí. Oba jsou si velmi podobné s jediným rozdílem, že při přidání stromu za vnitřní uzel je vždy třeba odříznout následující periody. Toto odřezávání je tedy další využití pro reference v grafu ukazující na vrchní a spodní uzel (*NodeAbove* a *NodeUnder*). V jednom průchodu jsou nejprve všechny zpětné reference na novou poslední periodu vynulovány a následně jsou vynulovány také dopředné reference na odříznutou periodu. Díky tomu je zajištěno, že na odříznuté uzly žádný ukazatel neodkazuje, a je tak možné v případě potřeby uvolnit jimi alokovanou paměť. Po odříznutí period je pak už postup stejný pro oba přístupy.

Z původního uzlu je vytvořen kořen nového stromu a z něj je na základě vložených parametrů vygenerovaný strom reprezentující další opci.

5.2.5 GUI

Po implementaci algoritmů stanovujících hodnoty ve stromě bylo třeba přejít k vrstvě grafického rozhraní (GUI) tak, aby uživateli přehledně zobrazovala průběhy výpočtu a jednotlivé pohledy na binomický model. Finální verze GUI základní obrazovky implementované aplikace je vidět na obrázku 5.1 a její zásadní části jsou popsané prvek po prvku v rámci této sekce. Volba prvků i jejich rozložení vychází z načrtlého návrhu, který byl popisován v kapitole 4.3.6. Další okna aplikace se využívají pro nastavení nebo třeba vyplňování konverzní tabulky.



Obrázek 5.1: GUI implementované aplikace

Pro pozicování jednotlivých prvků grafického rozhraní byla využita mřížka složená z menších mřížek (ve WPF *Grid*). Vrchní menu tvoří první řádek, plocha pro vykreslování stromů leží přes následující dva řádky. Parametry jsou umístěné ve vlastní mřížce, která leží ve třetím řádku. Z vertikálního pohledu je posuvník pro zvětšování a zmenšování stromu umístěný v prvním

sloupci, vykreslovací plocha ve druhém sloupci a parametry pak ve sloupci třetím.

Vykreslování binomického stromu

Zásadním prvkem celého grafického rozhraní je plocha pro vykreslování vývoje binomického stromu. Při implementaci navrženého způsobu zobrazování bylo třeba zvolit takový panel, který na sebe umožní napozicovat další objekty. Jako nejvhodnější se ukázal prvek z WPF, který se nazývá *Canvas* (neboli plátno). Důvodem pro zvolení tohoto panelu byla jeho schopnost pozicovat vkládané prvky pomocí odsazení shora (top) a zleva (left). Tento přístup je ideální pro vykreslení stromu do zmiňované pomyslné tabulky.

Po zvolení vhodného plátna bylo třeba vybrat prvek, který bude vhodně reprezentovat uzel ve stromě. Rozhodování probíhalo mezi klasickým tlačítkem (*Button*), jednoduchým obdélníkem (*Rectangle*) a vyplněným rámečkem (*Border*). Nejvhodnějším prvkem se od začátku jevilo tlačítko, které bylo nakonec také použito. Důvodem pro zvažováním zbylých dvou prvků byla pouze snaha urychlit počáteční vytváření stromu, které při větších instancích způsobuje krátkou prodlevu.

Mezi velké výhody tlačítka patří beze sporu možnost přímo mu nastavit text (*Content*) a kontextové menu. Zároveň je tlačítko vhodným grafickým prvkem a lze ho v případě potřeby deaktivovat. *Border* je tlačítku velmi blízký, také poskytuje možnost zvolit *Content*, použít kontextové menu a tak dále. Jeho použití bez dalších úprav dokonce představovalo i mírné navýšení rychlosti, po manuálním přidání chybějících grafických prvků se ale *Border* ukázal být ještě pomalejší, než původní varianta. Tvar obdélníku (*Rectangle*) byl na vykreslování suverénně nejrychlejší. Zásadní problém byl ale v tom, že se jedná o primitivní objekt, u kterého nelze nastavit ani vnitřní text, natož s ním provádět nějaké složitější operace. Po manuálním přidání vrstvy textu přes vygenerovaný tvar se navíc rychlost vykreslování opět přiblížila hodnotám u tlačítka. Přes snahu najít lepší variantu bylo tedy nakonec jako element reprezentující uzel použito tlačítko. Samotné vykreslování probíhá tak, že se při prvním vykreslení nezměněného stromu vytvoří pro každý uzel instance ViewModelu (VM) a ta se uloží do speciální kolekce vhodné pro *Data Binding*. Při vytváření VM je specifikovaná jeho pozice, šířka a výška tak, aby se do něj vešel obsah, tedy hodnota uzlu pro vybraný pohled na strukturu.

V požadavcích i návrhu bylo řešeno, že je třeba implementovat skrývání stromů, které by se jinak překrývaly se svými sousedy. Pro tyto elementy, které jsou součástí skrytého stromu, jsou ViewModely vytvořeny také a to s jediným rozdílem, nesou si s sebou příznak o tom, že nemají být viditelné. Vykreslování a hlavně vytváření objektů reprezentující uzly s každou změnou od znovu se totiž ukázalo být velmi pomalé. Vytvoření a vykreslení relativně komplikovaného objektu jako je tlačítko trvá, tento přístup však reakční dobu u častých akcí, jako je například skrývání a zobrazování stromů, výrazně zlepšil.

Po vytvoření VM pro každý z uzlů je vyvolána událost, která notifikuje

prvky svázané s kolekcí o nutnosti překreslit svůj obsah. Vytvoření mnoha prvků v jednom panelu je zajištěno díky použití prvku *ItemsControl*. Jedná se o speciální element, který umožňuje zobrazit celou kolekci objektů.

Různé pohledy na strom

Pro zobrazení všech a zároveň jen jednoho vybraného pohledu na rozvoj hodnot ve struktuře byl zvolen prvek *TabControl*, který umožňuje rozdělit několik pohledů do tabů, které jde snadno přepínat. Pro někoho je výhodou, že je standardně obsah tabů mezi sebou zcela nezávislý. Při použití *TabControlu* v implementovaném nástroji to však byla zásadní nevýhoda. Při přepínání tabů totiž docházelo k opětovnému vykreslování a vytváření celého obsahu několikrát, což velmi zpomalovalo odezvu při práci s aplikací. Tento problém však byl vyřešen pomocí přetížení šablony pro vykreslování obsahu prvku. Díky tomu je ve finální verzi aplikace sdíleno jen jedno plátno napříč všemi taby. Jediné, co se tedy při změně tabu mění, je obsah jednotlivých uzlů. Všechno ostatní zůstává stejné, nic se znovu nevytváří, ani nemaže.

Pro případy, kdy by chtěl uživatel vidět všechny ukazatele pro daný uzel současně, bylo implementováno také detailní zobrazení, které kombinuje všechny hodnoty vztahující se k vybranému uzlu. Tento detail využívá WPF prvku *Popup*, což je vlastně jen mřížka překrývající aktuální okno naplněná hodnotami vztahující se k danému uzlu.

Manipulace se stromem

Protože můžou nastat situace, kdy je skutečná velikost plátna mnohem větší, nebo naopak menší, než je velikost obrazovky, bylo třeba najít způsob, jak plátno na žádost uživatele zmenšit, zvětšit, nebo posunout. Pro tyto účely byly zvoleny dva prvky.

Prvním je *ScrollViewer* (ve volném překladu posuvný hledáček), ve kterém je vložený celý panel pro zobrazování. *ScrollViewer* totiž na základě velikosti svých potomků, v tomto případě panelu *Canvas*, v případě potřeby zobrazí posuvníky, pomocí kterých lze náhled posunout.

Druhým elementem je *ScaleTransform* (ve volném překladu manipulátor měřítka). Jak už název napovídá, *ScaleTransform* umožňuje dynamicky měnit měřítko náhledu a tedy i jeho aktuální velikost.

Zadávání parametrů

Po vykreslení základních modelů pro pevně zakódované parametry v pracovní verzi aplikace bylo třeba zprovoznit formulář s textboxy svázanými s parametry nutnými pro výpočet. Už v prvním návrhu byly parametry umístěny po pravé straně od prostoru pro graf. Ukázalo se ale, že v určitých případech není nutné parametry zobrazovat a ubírat tak prostoru pro výsledky. Z tohoto důvodu byl použitý WPF prvek *Expander*. *Expander* totiž umožňuje v případě potřeby buď skrýt, nebo naopak rozšířit svůj obsah. Díky tomu lze parametry skrýt a soustředit se pouze na výstupy.

Nastavení

Tlačítko pro otevření okna umožňující měnit konfiguraci aplikace bylo umístěno do horní lišty menu. Po otevření lze nastavení měnit. To je pak po stisknutí tlačítka *Save* dále ukládáno do konfiguračního souboru aplikace. Díky tomu je zajištěno, že se nastavení nezmění ani po restartování nástroje. Po jeho startu totiž dochází k načítání tohoto souboru. V případě, že je soubor nedostupný, jsou použité defaultní hodnoty.

5.2.6 Ukládání snímků obrazovky

Jedním z požadavků na aplikaci byla také možnost ukládat snímky obrazovky, respektive vykresleného grafu. Tato funkcionalita byla implementována dvěma způsoby.

První způsob umožňuje uložit aktuálně vykreslené plátno jako obrázek ve formátu PNG. Jedná se o velmi populární bezztrátový formát určený pro kompresi rastrové grafiky. Jeho použití se tedy zdálo více než vhodné, problém ale nastal při exportu do něj. Při exportování obsahu panelu do PNG je totiž v rámci frameworku .NET nutné nejprve vykreslit panel do bitmapy bez jakékoliv komprese a následně jí teprve zkompresovat a převést do cílového formátu. Z tohoto důvodu je formát PNG nevhodný pro velké instance stromů, protože se bitmapa využívaná jako mezikrok jednoduše nevejde do paměti.

Z důvodu paměťové náročnosti způsobené vykreslováním bitmapy byl implementovaný ještě jeden formát a tím je XPS. XPS na rozdíl od PNG není grafický formát, ale formát určený pro reprezentaci dokumentů. Slouží jako formát tiskové fronty a je velmi podobný formátu PDF. Důvodem pro uvedení požadavku na ukládání snímků obrazovky byla především možnost tisknout generované výstupy, a k tomu je XPS přímo určený. Použitá technologie WPF ho navíc stejně jako operační systém Windows podporuje již od roku 2007, tedy od Windows Vista, kdy bylo WPF poprvé představeno. [26]

5.2.7 Reporting

Generování vhodných reportů bylo implementováno na základě navrženého algoritmu v sekci 4.3.7. Jako pomocná proměnná obsahující již vytisknuté stromy byl pro efektivitu zvolený *HashSet*, tedy množina s konstantním složitostí vkládání a získávání prvků.

Zajímavostí při implementaci této části byla volba technologie. První verze totiž byla implementována pomocí knihovny *Microsoft Excel Interop* [27]. Tato volba se však ukázala být velmi nešťastná. Knihovna totiž funguje tak, že v kódu zpřístupní API rozhraní nainstalovaného balíku Microsoft Office. Zkratka API přitom označuje *Application Programming Interface*, tedy rozhraní pro programování aplikací. Díky tomu se jedná o opravdu mocný nástroj s širokou paletou funkcí. Bohužel s tím ale souvisí také zásadní nevýhody. V prvé řadě totiž knihovna nefunguje na počítači, kde není nainstalovaný *Microsoft Office*. Tento nedostatek by šel teoreticky omluvit tím, že nedává smysl generovat reporty pro program, který uživatel nemá nainstalovaný. Ještě

zásadnějším problémem je ale rychlost knihovny. V situacích, kdy je cílem programátora vzít matici a vytisknout jí do souboru, nevzniká žádný problém, protože lze cíle dosáhnut pomocí jednoho volání API. Pro případy, kdy je nutné volat API pro vytisknutí každého uzlu, jako je tomu třeba v implementovaném nástroji, se ale *Microsoft Interop* vyloženě nehodí. Samozřejmě by šlo vytvořit pomocnou matici, do které by byla struktura nejprve převedena a následně jedním voláním vytisknuta, jistě ale existuje i lepší řešení.

Vlastní implementace exportu do xlsx by byla zdlouhavá a náchylná na chyby. Lepším řešením bylo v tomto případě myšleno využití knihovny třetí strany, konkrétně knihovny *ClosedXML* [28]. Ta je distribuovaná pod open-source licencí MIT prostřednictvím balíčkovacího systému přímo ve vývojovém prostředí *Visual Studio*. Syntaxe *ClosedXML* je přitom velmi blízká knihovně Interop, paleta funkcí je pro účely této práce více než dostačující, pro fungování exportu není třeba mít nainstalovaný *Microsoft Office*. Generování reportů přitom trvá jen zlomek času předchozího řešení. Například pro instanci o 25 periodách trvalo generování kompletního reportu při volání API MS Office zhruba 30 s. Při využití *ClosedXML* export stejných dat trval pouze několik desítek ms (dle využití procesoru). Z tohoto důvodu bylo první řešení vypuštěno a ve finální verzi se o generování reportů stará zmíněná knihovna třetí strany.

Aby byla metoda pro generování reportu dostatečně obecná, bylo v ní využito tzv. delegáta vlastnosti uzlu. Jedná se v podstatě o proměnnou, ve které je uložený ukazatel na Getter zvolené vlastnosti (*Property*) třídy *Node*. Tento ukazatel lze volat jako klasickou metodu, jejíž parametr je instance uzlu určená k vytisknutí. Metoda pak vrací hodnotu vybrané vlastnosti. Tedy například při tisku vývoje vnitřních hodnot je parametrem metody delegát vlastnosti *InnerValue*.

5.2.8 Ukládání do souboru

Jedním z posledních zásadních požadavků, který bylo třeba implementovat bylo ukládání rozdělané práce do souboru. Pro uložení opravdu celého stavu aplikace, bylo třeba serializovat celou instanci třídy *RealOptionProject* a všechny její vlastnosti. Při implementaci této funkcionality se nabízely dva přístupy.

První přístup spočíval ve vlastní implementaci ukládání do souboru. Šlo by tedy o postupné ukládání parametru po parametru, další průchod stromem a pak opětovné načítání obráceným způsobem. Při ukládání i načítání by navíc bylo třeba dodržet všechny reference, kterých je ve struktuře mnoho, včetně těch cyklických. Z tohoto důvodu byla tato varianta brána pouze jako záložní řešení pro případ, že by podobný mechanismus nebyl ve frameworku už implementován.

Ukázalo se, že takový mechanismus ve frameworku opravdu existuje. Jedná se o třídu *DataContractSerializer*, která slouží právě k serializaci a deserializaci komplexních objektů, jakým navržená struktura je. Pro zahrnutí objektů do ukládaného souboru stačí ke každé deklaraci třídy a jejím vlastnostem přidat vhodný atribut, zavolat patřičnou metodu se správným nastavením a .NET

se o ukládání i načítání postará. Správným nastavením je v tomto případě myšleno především zapnutí zachování referencí, díky kterému nejsou instance při ukládání vytvářeny několikrát, ale pouze jednou a v dalších případech jsou nahrazeny odkazem na ně. Cílovým formátem ukládaného souboru je standardní XML. [29]

5.2.9 Vlákna na pozadí

Na závěr implementační fáze byly některé výpočty a náročnější operace přesunuty z hlavního vlákna do vlákna běžícího na pozadí. Pro tyto účely se ve frameworku využívá třídy zvané *BackgroundWorker* (v doslovném překladu pracovník na pozadí). Při využívání pouze jednoho vlákna totiž hrozí, že při náročnějších operacích aplikace z pohledu uživatele zamrzne a přestane reagovat. Ona ve skutečnosti pouze provádí složitější operaci, jakou může být například vytváření reportu, generování stromu, nebo konverze hodnot IT investice.

V takových případech je díky vykonávání práce na pozadí přes grafické rozhraní zobrazený indikátor, který informuje uživatele o tom, že právě probíhá nějaký výpočet a je třeba chvíli počkat. Ve většině případů se jedná pouze o zlomek sekundy, pro opravdu velké stromy se však interval může mírně zvýšit, je tedy lepší uživatele informovat.

Bohužel do vedlejšího vlákna nelze přesunout všechny dlouho trvající operace. Největším problémem aplikace je totiž *Data Binding* velkého množství uzlů. Aktualizaci GUI lze totiž provádět pouze z hlavního vlákna. Pozicování jednotlivých uzlů prostřednictvím vytváření jejich ViewModelů při průchodu stromem tím pádem sice může probíhat ve vedlejším vlákně, aktualizace panelů už ale probíhá ve vlákně hlavním, což může v určitých případech způsobovat mírnou prodlevu při vykreslování. S vedoucím práce však bylo domluveno, že zásadní jsou výstupy aplikace, nikoliv to, jestli vykreslení stromu o 50 úrovních trvá sekundu, nebo sekundu a půl. Takto velké instance totiž stejně nejsou standardním vstupem.

5.3 Nasazení

Jedním z nefunkcionálních nároků na navržený nástroj byla jednoduchost nasazení, kterou zvládne i méně zkušený uživatel počítače. Z tohoto důvodu byla pro vytvoření instalačního balíčku zvolena technologie *ClickOnce*. Tato technologie nasazení umožňuje vytvářet aplikace pro operační systém Windows, které jsou snadné na spuštění i instalaci. Visual Studio navíc poskytuje plnou podporu pro publikování a aktualizaci aplikace nasazené touto technologií. Aplikace publikované touto technologií lze dokonce instalovat bez administrátorského oprávnění. [30]

Předpokladem pro správnou funkčnost aplikace je mít v počítači operační systém Microsoft Windows a framework Microsoft .NET Framework ve verzi 4.6 a vyšší. Jedná se o verzi, která je běžně dostupná ve Windows 10, do starších verzí jí lze doinstalovat. Součástí instalačního balíčku je přeložený

nástroj pro 32-bitový i 64-bitový operační systém. Stručný návod, jak aplikaci nainstalovat, je součástí instalačního balíčku (soubor readme.txt).

5.4 Zhodnocení implementace

Implementace nástroje pro podporu výpočtu hodnoty reálných opcí byla úspěšná. Veškeré funkcionální i nefunkcionální požadavky vycházející ze zadání i z pravidelných konzultací s vedoucím práce byly splněny. Po dokončení této části je proto možné přejít ke kapitole věnující se testování implementovaného nástroje.

Kapitola 6

Testování

Na základě analýzy doporučené literatury a prací na podobné téma byla stanovena metodika testování. Bylo zjištěno, že existující práce nejsou pro otestování implementovaného nástroje úplně vhodné, protože nereflektují celý průběh výpočtu od obecné části až po výpočet hodnoty IT investice. Rovněž neexistuje obecný vzorec pro provedení transformace z vývoje spotové ceny do cen za cloud a on-premise. Z tohoto důvodu byla po dohodě s vedoucím práce při testování zvlášť ověřována obecná část výpočtu a část věnující se hodnotám IT investic. Aby bylo možné nástroj testovat tímto způsobem, bylo třeba zanedbat vztah mezi hodnotou spotové ceny pro danou úroveň a poplatky za cloud a on-premise prostředky v dané úrovni. Díky tomuto zjednodušení bylo možné oba dva výpočty ověřovat nezávisle, aniž by byla ovlivněna správnost výpočtu. Z toho navíc plyne, že se jedná o univerzální použití, které lze aplikovat i na jiné úlohy podobného charakteru.

Kapitola jako taková je rozdělena do tří částí. První dvě se věnují testování správné funkčnosti aplikace, třetí je věnována experimentům s vývojem IT investic. V první sekci jsou porovnávány výstupy aplikace s výsledky příkladů z literatury a z nástroje pro *Microsoft Excel* pro obecné výpočty týkající se hodnoty reálné opce. Testování specifických výpočtů hodnot IT investic je pak věnována část druhá. V ní jsou výstupy porovnávány přímo s předpisem pro výpočet vývoje prvních pěti period, který je součástí přiloženého CD jakožto soubor se vzorci pro *Microsoft Excel*. Poslední, třetí část se pak zabývá experimentálním testováním týkající se hodnot IT investic. V rámci těchto testů byly vytvořeny různé scénáře, pro které byla do aplikace dosazována vstupní data, která způsobovala různé reakce na výstupu. Tyto reakce jsou pak v kapitole dále diskutovány.

Celá kapitola je psaná stručně a slouží spíše jako popis složek *testy* a *experimenty* na přiloženém CD, ve kterých je pro každý uvedený test vytvořená složka obsahující vygenerovaný report a stav aplikace, který lze do ní opětovně načíst. Součástí každého testu ověřující správnost výpočtu jsou také referenční data, se kterými lze výstupy porovnat a ověřit.

6.1 Správnost obecného výpočtu

V první části testování bylo třeba ověřit, že výstupy pro příklady obecných reálných opcí, které implementovaná aplikace generuje, jsou korektní. K tomu bylo využito několik různých příkladů z literatury, ze kterých byla vygenerována vstupní data. Konverzní tabulka související s výpočtem hodnoty IT investice byla při těchto testech vyplněná nulami. Konkrétní použité vstupy jsou pak uvedené v tabulkách 6.1 a 6.2.

Konkrétně byly provedeny tyto testy:

- *sec_3_3_1* – tento test spočíval v dosazení hodnot z příkladu 5.2 z literatury [6], který byl blíže popsán v sekci 3.3.1,
- *sec_3_3_2* – data pro tento test pochází z příkladu 5.3 z [6], který byl přiblížen v sekci 3.3.2,
- *XLS_Scholl_1* – tento test spočíval v ověření výstupů proti nástroji pro MS Excel, který byl součástí přiloženého CD u knihy [6],
- *XLS_Scholl_2* – výstupy tohoto testu byly také ověřovány proti nástroji pro MS Excel z [6],
- *Nosk_str40* – vstupní data pro tento test pochází ze strany 40 diplomové práce [11],
- *Nosk_str47* – také data pro tento test pochází z diplomové práce [11], v tomto případě ze strany 47.

Test	sec_3_2_1	sec_3_2_2	XLS_Scholl_1	
typ opce	kupní	prodejní	kupní	prodejní
S	200 000	10 000 000	17 000	
X	100 000	8 000 000	15 000	
T	6	5	7	
n	6	5	14	
σ^2	0,4	0,6	0,55	
r	0,045	0,038	0,06	
Výsledek	ok	ok	ok	ok

Tabulka 6.1: Testování správnosti obecného výpočtu (1. část)

Pomocí těchto příkladů bylo ověřeno, že první, obecná část výpočtu funguje správně. Z pohledu aplikace se tedy jednalo o ověření správnosti vývoje zobrazovaného na prvních třech tabech (*Current Price*, *Inner Value*, *Real Option Value*) a na tabu posledním (*IT Investment Option Value*), který využívá stejného algoritmu pro zpětný průchod jako třetí tab (*Real Option Value*).

Funkčnost aplikace byla dále průběžně ověřována v průběhu celé implementační i testovací fáze na různých dalších příkladech, které zde pro svou podobnost s popsanými daty uvedené nejsou.

Test	Nosk_str40	Nosk_str47	XLS_Scholl_2	
typ opce	kupní	kupní	kupní	prodejní
S	10 895 357	43 335 222	50 000	
X	20 000 000	20 000 000	70 000	
T	2	2	3	
n	8	8	10	
σ^2	0,6	0,6	0,81	
r	0,04	0,04	0,04	
Výsledek	ok	ok	ok	ok

Tabulka 6.2: Testování správnosti obecného výpočtu (2. část)

6.2 Správnost vývoje hodnoty IT investice

Ve druhé fázi testování bylo třeba ověřit správnost vývoje hodnoty IT investice. Protože se jedná o zcela nový přístup k hodnocení investice, neexistují k němu v tuto chvíli žádná referenční data v literatuře, ani na internetu. Z tohoto důvodu byl průběh výpočtu ověřován pouze obecně přímo proti předpisu pro prvních pět úrovní, který je součástí přiloženého CD. Předpis byl vytvořen společně s vedoucím práce a dále konzultován s prof. Starým.

Ve složce *xls* je uložený předpis, který je pro první čtyři úrovně vidět také na obrázku 3.9. Ve složce pojmenované dle názvu testu je pak vždy uložen další soubor, který už obsahuje implementované vzorce včetně dosazených hodnot pro příslušný test. S obsahem tohoto souboru lze pro každý test porovnat vygenerované výstupy a ověřit tak jejich správnost. V případě potřeby lze dále proměnné v souboru měnit a ověřit si jiné vstupy dle svého uvážení.

Pro ověření správnosti byly použity tyto hodnoty:

Na popsaných vstupních datech bylo ověřeno, že výpočet v aplikaci probíhá přesně dle určeného předpisu. Test byl sice provedený pouze pro prvních pět úrovní, ale taková vstupní instance je dostatečně velká k tomu, aby ověřila všechny situace, které mohou při výpočtu nastat (viz 3.4).

Funkčnost druhé části výpočtu byla samozřejmě také testována v průběhu implementace. Vzhledem ke způsobu testování a podobnosti výpočtu pro různá vstupní data v této části další vstupy uvedené nejsou. Pro bližší zkoumání vývoje IT investice při různých situacích slouží následující sekce.

6.3 Experimentální testování

Jak již bylo řečeno v úvodu kapitoly, pro experimenty s hodnotami IT investic bylo po analyzování příkladů z literatury a po dohodě s vedoucím práce zvoleno několik scénářů, které se vždy soustředí na změny zvoleného vstupního parametru. Vstupní parametry vždy vychází z reálných hodnot, výjimku tvoří spotová a realizační cena a dále poplatky za on-premise a cloud. Tyto hodnoty byly voleny obecně. Tento přístup k experimentování byl zvolen pro možnost ověření vzorců pro různé situace na trhu a pro ověření vlastností cloudu.

Test	IT_Investice_1		IT_Investice_2	
úroveň	*anuita*	*cloud*	*anuita*	*cloud*
5	530	2000	3500	4500
4	480	1500	0	3500
3	250	1100	1700	3000
2	110	900	0	2000
1	320	800	540	1500
0	0	500	0	1200
-1	0	450	0	800
-2	0	400	0	500
-3	0	320	0	300
-4	0	290	0	250
-5	0	250	0	200
p	0,345541		0,373044	
d	0,654459		0,626956	
Výsledek	ok		ok	

Tabulka 6.3: Testování správnosti výpočtu hodnoty IT investice

Stanovení metodiky pro tyto odhady a z ní vycházející transformace spotové ceny na poplatky za infrastrukturu nebylo cílem této práce a je tak na každém uživateli aplikace, aby si taková data sestavil.

Následující podkapitoly této sekce reprezentují jednotlivé experimenty. Jejich součástí jsou tabulky obsahující vstupní data obecné části výpočtu a popis dat do výpočtu hodnoty IT investice. Konverzní tabulky obsahující všechny poplatky za cloud a on-premise jsou z důvodu obsáhlosti umístěny pouze v elektronické formě na přiloženém CD. V kapitole nejsou pro svou obsáhlost ani obrázky reprezentující vývoje hodnot v binomických stromech. Ke každému experimentu je ale na přiloženém CD vytvořená složka (v adresáři *experimenty*), která obsahuje vygenerovaný report ilustrující vývoje hodnot, konverzní tabulku a uložený stav aplikace, který lze do ní znovu načíst. Přímo ve v této složce je pak soubor pro *Microsoft Excel*, který obsahuje všechna vstupní data pohromadě (není možné ho načíst do aplikace, slouží pouze jako přehledný soupis).

V průběhu všech experimentů kromě posledního byla vypnutá možnost dopočítávat anuity za on-premise prostředky, které překročí životnost opce. Na tuto funkcionalitu se soustředí poslední experiment, který porovnává výstupy s a bez zapnuté funkce. Dopočítávání bylo ve většině případů vypnuté, protože výrazně ovlivňuje výsledky směrem nahoru, což může snižovat přesnost a zvýhodňovat cloud na úkor on-premise.

6.3.1 Různá volatilita

Prvním testovaným scénářem byl experiment s různou mírou volatility, tedy nejistoty trhu. Nízká volatilita představuje minimální riziko, používá se pro ustálený trh bez výkyvů. Vysoká volatilita naopak implikuje rychle se měnící

trh, kde je velká šance úspěchu, ale také neúspěchu. Scénář tedy testuje vlastnosti vzorců pro různé trhy.

Vstupní data

Tento experiment se zaměřuje na to, jak je konkrétní část výpočtu hodnoty IT investice ovlivněna změnou volatility v obecné části výpočtu. Konkrétní vstupní hodnoty do obecné části výpočtu ilustruje tabulka 6.4. Jak je vidět, bylo provedeno celkem šest testů. Tyto testy byly vždy prováděny pro 24 period ve dvou letech, jedno období tedy představovalo jeden měsíc. Bezriziková úroková míra byla zvolena 1 % na základě výnosnosti státních dluhopisů (viz [31]). A protože bylo cílem experimentu otestovat reakce na různou míru volatility, byla pro každý test zvolena jiná hodnota. První testy počítaly s nízkou volatilitou na ustáleném trhu, poslední testy pak už počítaly s vysokou volatilitou a tedy i velkým rizikem. Hodnoty volatility a jejich kategorizace vychází z diplomové práce na podobné téma [11]. Poplatky za cloud a on-premise jsou součástí přiloženého CD. Byly voleny tak, aby paušál za cloud zhruba odpovídal cenám on-premise. Pro výpočet hodnot anuit byla využita kalkulačka [32]. Fixní poplatek za navýšení ceny v cloudu byl zvolen 450 jednotek, standardní hodnota anuity byla zvolena 470 jednotek, což odpovídá nákupu prostředků v hodnotě 10 000 jednotek. Předpokládalo se také, že po každých deseti navýšení výkonu on-premise bude třeba provést větší investici do HW. Každá desátá anuita je proto vyšší a má hodnotu 1412 jednotek což představuje nákup za 30 000.

Test:	1	2	3	4	5	6
S	8000					
X	7000					
T	2					
n	24					
r	0,01					
σ^2	nízká	střední	vyšší		vysoká	
	0,1	0,25	0,4	0,5	0,7	0,9
typ	kupní (call)					
Anuita (běžná)	470					
Anuita (vyšší)	1412					
Změna cloud	450					
Hodnota r.opce	1966	2687	3189	3463	3921	4298
Hodnota IT inv.	229	245	257	264	277	289

Tabulka 6.4: Experiment 1 s různou volatilitou

Vyhodnocení

Jak ilustrují výsledky, které jsou na posledních dvou řádcích tabulky 6.4, s rostoucí volatilitou roste nejen hodnota reálné opce, což je standardní

chování, ale také hodnota IT investice. Tento výsledek byl předpokládaný a potvrdilo se tak, že čím vyšší je nejistota a hrozící rizika, tím vyšší přínosy plynou z využívání cloudové infrastruktury, která je flexibilní a dokáže se trhu aktivně přizpůsobovat.

6.3.2 Různá délka a počet období

Další experiment byl zaměřený na ověření reakcí na změny týkající se různě dlouhých životních cyklů a různých počtů období. I v tomto scénáři jedno období reprezentovalo jeden měsíc. Z toho plyne, že pro krátká období (např. čtvrt roku) bylo nutné splatit zakoupené on-premise vybavení výrazně rychleji, než pro dlouhá období (např. dva roky).

Vstupní data

Vstupní data vychází z dat pro předchozí experiment. Liší se zde ale počet období, délka cyklu a hodnoty anuit, které vždy souvisí s počtem období. Testovány byly relativně krátké cykly, první test byl provedený pro tři měsíce, tedy čtvrt roku. Následovaly testy pro půl, tři čtvrtě roku, rok, rok a půl a dva roky. Jedno období opět odpovídalo jednomu měsíci. Volatilita byla už zvolena standardně pro oblast IT, jak jí uvádí [11], tedy vysoká. Hodnoty anuit byly počítány opět pomocí kalkulačky [32]. Běžná anuita odpovídá standardnímu dokupování on-premise prostředků, byla určena z částky 10 000, vyšší anuita je využívána pro každý desátý nákup a odpovídá nákupu za celkem 30 000. Konkrétní hodnoty jsou uvedené v tabulce 6.5.

Test:	1	2	3	4	5	6
S	8000					
X	7000					
T	0,25	0,5	0,75	1	1,5	2
n	3	6	9	12	18	24
r	0,01					
σ^2	vysoká 0,6					
typ	kupní (call)					
Anuita (běžná)	3400	1725	1167	888	609	470
Anuita (vyšší)	-	-	-	2665	1829	1412
Změna cloud	450					
Hodnota r.opce	1757	2220	2545	2848	3319	3705
Hodnota IT inv.	949	516	385	329	282	270

Tabulka 6.5: Experiment 2 s různou délkou a počtem období

Vyhodnocení

Hlavní výstupy experimentu jsou umístěny v tabulce 6.5. V tomto scénáři se ukázalo, že s rostoucí délkou životního cyklu roste i hodnota reálné opce. Hodnota IT investice naopak lehce klesá. Tato vlastnost je způsobená tím, že při malém počtu období je finančně náročné splatit velké investice do on-premise. U cloudu tento problém nevzniká. Navyšování tarifu totiž není závislé na tom, na jak dlouho se tarif zvyšuje. Cloud je flexibilní, tedy lze výkon stejným způsobem další měsíc zase snížit. Toto však on-premise neumožňuje. Tato flexibilita cloudu se tedy při srovnání pro krátkou periodu projevuje více. Při větším počtu období hodnota IT investice klesá, protože se k sobě blíží ceny anuit a rozdíly v tarifech cloudu.

6.3.3 Občasné zvyšování on-premise

Tento scénář simuluje situace, kdy nejsou on-premise prostředky dokupovány každou periodu, ale pouze jednou za čas. Taková praxe může velmi snadno nastat i v reálném životě. Při prvním nákupu se pořídí prostředky s rezervou, která je využita při nutném navýšení výkonu. Ve chvíli, kdy už je výbava nedostatečná, dojde k většímu nákupu, který opět vydrží několik navýšení. Při testování tohoto scénáře byly dále použity dva přístupy.

První přístup počítá s dokupováním později. Jedná se tedy o situace, kdy je opravdu nejprve nakoupený předimenzovaný HW včetně potřebných licencí na SW a další upgrade se provádí až v případě potřeby. Druhý přístup naopak počítá s tím, že byla zvolená poddimenzovaná konfigurace, která je třeba navýšit hned v první úrovni. Oba přístupy využívaly stejná data s různým rozložením, což je umožňuje porovnat.

Vstupní data

Vstupní data pro tento scénář opět vychází z předchozích experimentů. Data pro obecnou část všech testů zůstávala stejná. Znovu bylo simulováno období dvou let rozdělené po měsících. Bezriziková úroková míra vycházela ze státních dluhopisů a volatilita byla vysoká. Co se v tomto případě výrazně lišilo byly výše a rozložení poplatků za jednotlivé úrovně v cloudu a on-premise. V první fázi experimentu (označeno A) bylo uvažováno zvyšování lokální konfigurace později. Konkrétně to bylo vždy po navýšení o x úrovní, kde x je uvedeno v tabulce na řádku označeném *Dokupování jednou za*. V druhé fázi (B) pak bylo uvažováno dokupování jednou za stejný počet úrovní s rozdílem, že byla konfigurace vylepšena hned v první úrovni a pak jí bylo možné x úrovní využívat bez dokupování. Hodnoty anuit odpovídají tomu, že když se vylepšuje konfigurace jednou za tři úrovně, je nutné navýšit výkon tak, aby stačil pro následující tři úrovně. Tedy je částka trojnásobná. V případě velkého nákupu (jednou za deset úrovní) je částka pouze dvojnásobná, protože je jí nutné ve 24 úrovních vynaložit jednou namísto dvakrát. Data jsou vidět v tabulce 6.6.

Test:	1	2	3	4	5	6
S	8000					
X	7000					
T	2					
n	24					
r	0,01					
σ^2	vysoká 0,6					
typ	kupní (call)					
Anuita (běžná)	470	940	1410	1880	2350	2820
Anuita (vyšší)	1412	2824	2824	2824	2824	2824
Změna cloud	450					
Dokupování jednou za	1	2	3	4	5	6
Hodnota IT inv. A	270	247	238	232	226	222
Hodnota IT inv. B	270	322	425	567	709	851

Tabulka 6.6: Experiment 3 s pouze občasným navyšováním on-premise

Vyhodnocení

Výsledky jsou stejně jako vstupy v tabulce 6.6. V rámci experimentu se ukázalo, že pokud uživatel musí navyšovat lokální konfiguraci hned v první úrovni (B), je pro něj cloud výhodnější. Je to dáno tím, že poplatky za HW jsou při navyšování výkonu, aby stačil pro více úrovní, vyšší. Situace, kdy je nutné hned v první úrovni navýšit výkon např. pro následující 4 úrovně, aniž by to bylo v daný moment nutné, tak hrají lépe pro cloud, u kterého je tarif měněný rovnoměrně.

6.3.4 Vývoj ceny cloudu

Další, v pořadí čtvrtý, simulovaný scénář se soustředí na rozdíly cen mezi on-premise a cloudem. Během něj bylo provedeno šest testů, v rámci prvních třech testů byly poplatky za on-premise sníženy proti datům z předchozích experimentů, ve druhé polovině testů pak byl zdražen cloud.

Vstupní data

Obecné parametry byly voleny stejně jako v předchozích experimentech. Co se zde lišilo jsou poplatky za infrastrukturu. Jak již bylo řečeno v úvodu, v prvních třech testech byly poplatky za on-premise nižší než za cloud. V prvním testu byla anuita proti rozdílu úrovní cloudu zhruba poloviční, pak postupně rostla až do hodnoty z předchozích experimentů, které dosáhla v testu 3. V testech 4–6 pak cena za zvýšení tarifu v cloudu postupně rostla, až dosáhla několikanásobně vyšší hodnoty v testu 6. Data jsou v tabulce 6.7.

Test:	1	2	3	4	5	6
S	8000					
X	7000					
T	2					
n	24					
r	0,01					
σ^2	vysoká 0,6					
typ	kupní (call)					
Anuita (běžná)	225	370	470	470	470	470
Anuita (vyšší)	706	1312	1412	1412	1412	1412
Změna cloud	450	450	450	600	700	3000
Hodnota IT inv.	235	255	270	337	382	1468

Tabulka 6.7: Experiment 4 s rostoucí cenou cloudu

Vyhodnocení

Závěry plynoucí z tohoto experimentu nejsou tak jednoznačné jako v minulých případech. V testech 1–3 byla sice cena on-premise nižší, ale postupně rostla. To je vidět také na výsledcích. S rostoucí cenou on-premise se totiž zvyšuje také výhodnost cloudu, což se projevilo právě mírným růstem celkových výsledků, které si jsou velmi blízké.

Od testu čtyři pak už dále rostla pouze cena za cloud, ale pokračoval také růst celkového výsledku. Tento růst plyne z flexibility, kterou přináší cloud. Při snížení výkonu v cloudu se totiž dle použité metodiky šetří peníze za snížení tarifu. Výsledek je tak ovlivněn zejména členy ve spodní části stromu, kde se přičítá rozdíl aktuální úrovně vůči nulté úrovni. Tento rozdíl může být v určitých případech opravdu vysoký (zejména u testu 6). Nízké poplatky za on-premise jsou tak převáženy a výsledek ztrácí na přesnosti. Je tedy důležité volit data tak, aby byla relevantní a řádově srovnatelná.

6.3.5 Dopočítávání životnosti on-premise

V rámci posledního experimentu byla testována funkcionalita, která umožňuje při dokupování on-premise přičíst diskontované anuity, které by byly jinak placeny až v době po vypršení opce. Tedy v periodách, které už v rámci testu nejsou dále sledovány. Tato funkcionalita je běžně vypnutá, protože výrazně ovlivňuje výstupy. Existují totiž i takové průběhy investic (cesty stromem), ve kterých se dokupují prostředky až v poslední periodě, což zapříčiní to, že je k danému uzlu připočtena téměř celá cena za zakoupený výrobek. Dle implementované metodiky jsou totiž i tyto poplatky součástí celkové sumy, kterou uživatel získá s využitím cloudu. Pro srovnání byla použita stejná vstupní data jako v předchozích experimentech.

Vstupní data

Jako vstupy byla pro srovnání použita stejná data, jako v předchozích experimentech. Řádek *Původně* tabulky 6.8 obsahuje číslo původního testu. Například test 1.2 odkazuje na experiment 1 (volatilita), test 2.

Test:	1	2	3	4	5	6
Původně:	1.2	2.4	3A.2	3A.4	3B.3	4.4
S	8000					
X	7000					
T	2	1	2	2	2	2
n	24	12	24	24	24	24
r	0,01					
σ^2	střední 0,25	vysoká 0,6				
typ	kupní (call)					
Původ. hodnota	245	329	247	232	425	337
Hodnota IT inv.	282	492	339	322	735	355
Rozdíl	37	163	92	90	310	18
v %	15	50	37	39	73	5

Tabulka 6.8: Experiment 5 s dopočítáváním životnosti on-premise

Vyhodnocení

Během experimentu se potvrdilo, že dopočítávání anuit opravdu zvyšuje hodnotu investice. Jedná se o předpokládaný výsledek už z principu, co toto dopočítávání představuje. Jde totiž o přičítání v rámci experimentu nevyužité hodnoty on-premise, kterou uživatel teoreticky s využíváním cloudu získá nazpět, respektive jí nikdy neztratí. Nárůst výsledků byl se pohyboval v řádu jednotek až desítek procent. Nejvyšší rozdíl byl například zaznamenaný u testu z experimentu týkající občasného přikupování on-premise prostředků s variantou dokupu hned v první periodě. Tento výsledek byl způsoben vysokou cenou anuit, kterých nebylo v původním výpočtu započítáno mnoho. Jejich počet se ale s přičítáním nesplaceného dluhu více než zdvojnásobil, což způsobilo takto výraznou reakci na výstupu. Pokud by tedy byla brána v potaz nevyužitá kapacita lokální konfigurace po skončení experimentu, pak by cloudové řešení vycházelo ještě výrazně lépe než v původním experimentu.

Kapitola 7
Vyhodnocení

Cílem této práce bylo seznámit se s teorií reálných opcí a aplikovat jí na podpůrný nástroj, který bylo třeba navrhnout, implementovat a otestovat. Nástroj se měl zaměřit především na binomický model a na příklady z oblasti IT investic. Zásadní podmínkou úspěchu bylo vhodně navrhnout datovou strukturu reprezentující binomický strom. K ní bylo třeba vymyslet jednotlivé algoritmy a průchody touto strukturou, pomocí kterých byly generovány výstupy. Ukázalo se, že část věnující se oblasti IT investic byla výrazně obsáhlejší než bylo původně očekáváno, byl na ní proto kladen větší důraz. Přesto se všechny cíle podařilo splnit. Nástroj byl navržený tak, aby splňoval všechny definované požadavky. Následně byl také implementovaný a otestovaný. Testy přitom pokryly jak ověření správnosti výpočtu, tak různé experimenty, které potvrdily, nebo naopak vyvrátily různé úvahy nad implementovanou metodikou.

Mezi hlavní přínosy patří především přehlednost a jednoduchost navrženého nástroje. Zejména v porovnání s běžně používanými nástroji pro výpočty hodnot reálných opcí. Pokud by někoho omezovaly implementované funkce, je možné výstupy vyexportovat do sešitu pro *Microsoft Excel* a dále s nimi pracovat.

Co se ukázalo být velkým problémem bylo získávání vhodných vstupních dat, které by najednou ověřily celý průběh výpočtu. Problém nastal zejména v části týkající se IT investic. Protože se jedná o novou metodiku, o které teprve pojednává vedoucí práce ve své disertační práci [15], taková data zatím neexistují. Aby bylo možné aplikaci využívat pro komplexní příklady, musela by taková data být snadno získatelná i pro příklady z praxe, aktuálně tomu tak ale bohužel není. Jinak řečeno by musela existovat obecná transformace, která by dokázala převést hodnoty spotové ceny na poplatky za cloud a on-premise. S takovou transformací nepřímo počítá i navržený nástroj, který pro vyplnění konverzní tabulky mapuje jednotlivé úrovně na hodnoty spotové ceny. Taková transformace však zatím nebyla definována. Aktuálně se krom vedoucího práce této problematice věnuje další student ve své diplomové práci, který metodiku rozebírá více teoreticky, je tedy možné, že výstupem jeho práce bude mimo jiné tato transformace, kterou by bylo možné využívat pro vytváření vstupních dat. Problémy s daty však byly pro účely této práce vyřešeny rozdělením testování na obecnou část a na část z oblasti IT investic.

Obecná část byla testována na vstupních datech vygenerovaných z literatury, konkrétní část byla pak testována přímo proti předpisu od vedoucího práce. Experimentální testování probíhalo na jednotlivých scénářích, na kterých se potvrdily zajímavé vlastnosti metodiky.

Do budoucna by šlo určitě vylepšit několik věcí, v rámci aplikace by šlo zejména o rychlost vykreslování stromu. Proti první pracovní verzi byla sice rychlost výrazně vylepšena, použitý Data Binding ve spojení s tlačítky však v technologii WPF není nejrychlejším řešením. Nejedná se sice o zásadní nedostatek, ani omezující vlastnost, obzvláště pro malé vstupní instance, které tvoří většinu standardních vstupů, přesto by šlo na prostředí v budoucnu zapracovat. Z pohledu funkcionality navržené aplikace by bylo vhodné přidat funkce pro usnadnění generování vstupních dat. Přidána by mohla být například funkcionalita pro automatické počítání anuit pro jednotlivé úrovně z celkové výše investice. Případně by se nabízelo implementovat další přístupy k hodnocení IT investic, které vyplynou z hotových prací na toto téma.

V současné době je tedy krom tohoto textu výstupem funkční aplikace implementovaná ve frameworku .NET, která je vhodná k provádění různých experimentů s reálnými opcemi a s IT investicemi. Své využití proto najde především v akademické sféře s tím, že je otevřená dalšímu vylepšování. Po definování a doplnění zmíněné transformace spotové ceny na poplatky za on-premise a cloud by si své využití mohla najít také ve firmách snažících se odhadnout své budoucí náklady na infrastrukturu. Podobné grafy ilustrující vývoj investice by jistě ocenili také zákazníci cloudových poskytovatelů, na jejichž webu by tyto grafy mohly doplnit webové kalkulátory srovnávající on-premise a cloud.

Kapitola 8

Závěr

Cílem této práce bylo proniknout do problematiky reálných opcí a aplikovat získanou teorii při návrhu, implementaci a otestování podpůrného nástroje pro výpočty s ní spojené. Důraz měl být kladen hlavně na binomický model a na s ním související binomické stromy. Stromy mělo být možné různě procházet, měnit a generovat z nich výstupy. Aplikace měla krom výpočtů podporovat také generování reportů, ukládání rozdělané práce do souboru a několik dalších funkcí. Nástroj měl být otestován a experimentálně ověřen zejména na příkladech srovnávajících cloud a on-premise. Tento, i všechny ostatní stanovené cíle a požadavky se podařilo splnit a ověřit.

Nejprve byla zpracována nutná teorie týkající se opcí. Byly rozebrány finanční opce, reálné opce i vzorce pro srovnání cloudu a on-premise. Byly definovány jednotlivé typy opcí a z nich vycházející způsoby výpočtu. Ty byly pak následně zahrnuty do návrhu podpůrného nástroje, který byl posléze také implementován a otestován. Při testování bylo zvoleno několik přístupů tak, aby byla ověřena jak správnost generovaných výstupů, tak jejich zajímavé vlastnosti. Zaznamenaný průběh návrhu, implementace i testování je obsahem druhé, praktické části práce. Zaznamenaná data v elektronické podobě jsou součástí přiloženého CD.

Výstupem práce je krom tohoto textu funkční aplikace implementovaná ve frameworku .NET, která si své uplatnění najde především při provádění různých experimentů v akademické sféře. Aplikace je přehledná, jednoduchá a snadno nasaditelná. Po dokončení závěrečných prací [15][33] týkající se stejné problematiky a doplnění z nich plynoucích postupů by si své uplatnění mohla najít také v reálné praxi při zvažování IT investic. Jako pokračování této práce se tedy nabízí témata věnující se metodice týkající se určení vstupních dat, a transformaci spotové ceny na poplatky za cloud a on-premise.

Pro mě byla práce velkým přínosem hlavně díky seznámení se s pro mě novou technologií WPF, kterou jistě využiji i v praxi. Navrhování algoritmů pro oblast IT investic bylo pro mě zajímavou a poučnou zkušeností. Získané vědomosti uplatním zejména v profesním životě, hodnotím je tedy rozhodně kladně.

Literatura

[1] Weill, P.: The role and value of information technology infrastructure: some empirical observations. In *Strategic information technology management*, IGI Global, 1993, s. 547–572.

[2] Trnka, V.: *Možnosti budování firemní IT infrastruktury v malých a středních firmách s využitím cloudů*. Bakalářská práce, ČVUT Praha, Praha, 2015.

[3] Chellapp, R.: Intermediaries in Cloud-Computing: A New Computing Paradigm. [přednáška], Dallas: INFORMS Meeting, 1997.

[4] Peter Mell, T. G.: The NIST Definition of Cloud Computing. *NIST* [online]. září 2011 [cit. 2017-04-14]. Dostupné z: `http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf`

[5] Bezpalec, P.: Výhody a nevýhody Cloud computingu. *Publi* [online] [cit. 2017-04-16]. Dostupné z: `https://publi.cz/books/230/07.html`

[6] Scholleová, H.: *Hodnota flexibility. Reálné opce*. C. H. Beck, první vydání, 2007, ISBN 978-80-7179-735-7.

[7] Diskontování, diskontovaný. *AZ Data* [online]. 2015 [cit. 2017-04-18]. Dostupné z: `http://www.az-data.cz/slovnik/diskontovani-diskontovany`

[8] Schoellová, H.: *Investiční controlling: Jak hodnotit investiční záměry a řídit podnikové finance*. Grada Publishing a.s., první vydání, 2009, ISBN 978-80-247-2952-7.

[9] Ambrož, L.: *Oceňování opcí*. C. H. Beck, první vydání, 2002, ISBN 80-7179-531-3.

[10] Starý, O.: *Reálné opce*. A plus, první vydání, 2003, ISBN 80-902514-6-3.

[11] Nosková, V.: *Možnosti využití cloudů a reálných opcí pro začínající technologické firmy*. Diplomová práce, ČVUT Praha, Praha, 2016.

[12] Kislingerová, E.: *Manažerské finance*. C. H. Beck, třetí vydání, 2010, ISBN 978-80-7400-194-9.

[13] Mašát, M.: Analýza reálných opcí - nová metoda pro řízení projektů. *Ekonom.cz* [online]. 2016-01-23 [cit. 2017-04-23]. Dostupné z: http://ekonom.ihned.cz/c1-13821990-analyza-realnych-opci-nova-metoda-pro-rizeni-projektu

[14] Náplava, P.: Evaluation of Cloud Computing Hidden Benefits by Using Real Options Analysis. *Acta Informatica Pragensia* [online]. Vysoká škola ekonomická v Praze: Acta Informatica Pragensia, 2016 [cit. 2017-04-23], ISSN 1805-4951. Dostupné z: https://aip.vse.cz/index.php/aip/article/view/159/107

[15] Náplava, P.: *REAL OPTIONS AND THEIR USAGE IN INFORMATION TECHNOLOGIES*. Disertační práce (draft), ČVUT Praha, Praha, 2017.

[16] Chválová, J.: Co je anuita. *Peníze.cz* [online] [cit. 2017-04-23]. Dostupné z: http://www.penize.cz/slovnik/anuita

[17] Path-Dependent Option with Strong Memory. *investment-and-finance.net* [online]. 2013-04-27 [cit. 2017-04-14]. Dostupné z: http://www.investment-and-finance.net/derivatives/p/path-dependent-option-with-strong-memory.html

[18] Wolfram Mathematica. *Wolfram* [online] [cit. 2017-04-24]. Dostupné z: https://www.wolfram.com/mathematica/

[19] MATLAB. *MathWorks* [online] [cit. 2017-04-24]. Dostupné z: https://www.mathworks.com/products/matlab.html

[20] Real Options Valuation. *Real Options Valuation Inc.* [online] [cit. 2017-04-24]. Dostupné z: http://www.realoptionsvaluation.com/index.php

[21] Overview of the .NET Framework. *Microsoft Developer Network* [online]. 2017 [cit. 2017-04-30]. Dostupné z: https://msdn.microsoft.com/en-us/library/zw4w595w(v=vs.110).aspx

[22] Standard ECMA-334. *ECMA International* [online]. Červen 2006 [cit. 2017-04-30]. Dostupné z: http://www.ecma-international.org/publications/standards/Ecma-334.htm

[23] Introduction to WPF. *Microsoft Developer Network* [online]. Červen 2009 [cit. 2017-05-04]. Dostupné z: https://msdn.microsoft.com/en-us/library/mt149842(v=vs.110).aspx

[24] Dajbych, V.: MVVM: Model-View-ViewModel. *DOT NET Portál* [online]. 2009-04-21 [cit. 2017-04-30]. Dostupné z: http://www.dotnetportal.cz/clanek/4994/MVVM-Model-View-ViewModel

[25] The MVVM Pattern. *Microsoft Developer Network* [online]. 2017 [cit. 2017-04-30]. Dostupné z: https://msdn.microsoft.com/en-us/library/hh848246.aspx

[26] Krejčí, R.: Microsoft XPS - nová příležitost pro polygrafické provozy. *SvetTisku.cz* [online]. Říjen 2008 [cit. 2017-05-04]. Dostupné z: `http://www.svettisku.cz/buxus/generate_page.php?page_id=4601&buxus_svettisku=`

[27] Přístup k objektům Interop sady Office pomocí funkcí Visual C#. *Microsoft Developer Network* [online]. Červenec 2016 [cit. 2017-05-04]. Dostupné z: `https://msdn.microsoft.com/cs-cz/library/dd264733.aspx`

[28] ClosedXML on GitHub. *GitHub.com* [online]. Duben 2017 [cit. 2017-05-04]. Dostupné z: `https://github.com/closedxml/closedxml`

[29] Třída DataContractSerializer. *Microsoft Developer Network* [online]. 2017 [cit. 2017-05-04]. Dostupné z: `https://msdn.microsoft.com/cs-cz/library/system.runtime.serialization.datacontractserializer(v=vs.110).aspx`

[30] ClickOnce – zabezpečení a nasazení. *Microsoft Developer Network* [online]. 2017 [cit. 2017-05-04]. Dostupné z: `https://msdn.microsoft.com/cs-cz/library/t71a733d.aspx`

[31] Výnos desetiletého státního dluhopisu (maastrichtské kriterium) - ekonomika ČNB. *Kurzy.cz* [online]. 2017-04-30 [cit. 2017-05-06]. Dostupné z: `http://www.kurzy.cz/cnb/ekonomika/vynos-desetileteho-statniho-dluhopisu-maastrichtske-kriterium/`

[32] Annuity Payment (PV) Formula and Calculator. *Finance Formulas* [online]. 2017 [cit. 2017-05-06]. Dostupné z: `http://financeformulas.net/Annuity_Payment_Formula.html`

[33] Martin, M.: *Využití Realných opcí pro hodnocení investic cloudových technologií*. Diplomová práce (draft), ČVUT Praha, Praha, 2017.

Příloha A

Seznam použitých zkratek

API Application Programming Interface (rozhraní pro programování aplikací)

GUI Graphical User Interface (Grafické uživatelské rozhraním)

HW Hardware

IaaS Infrastructure as a Service (Infrastruktura jako služba)

IT Informační technologie

M Model

MS Microsoft

MVVM Model-View-ViewModel

NIST National Institute of Standards and Technology

PaaS Platform as a Service (Platforma jako služba)

SaaS Software as a Service (Software jako služba)

SLA Service Level Agreement

SW Software

V View

VM ViewModel

VS Visual Studio

Příloha B

Obsah přiloženého CD

- `popis.txt` ..... stručný popis obsahu CD
- `app` ..... implementovaná aplikace
  - `install_x86` ..... instalační balíček 32-bit
  - `install_x64` ..... instalační balíček 64-bit
  - `solution` ..... zdrojové soubory a projekt pro VS 2015
- `experimenty` ..... Záznamy z experimentálního testování
  - `1_volatilita`
  - `2_delky_obdobi`
  - `3a_obcas_zvys_pozdeji`
  - `3b_obcas_zvys_start`
  - `4_zmeny_cen_cloud`
  - `5_dopocitavani_anuit`
  - `experimenty.xlsx`
- `src` ..... zdrojová forma textu práce ve formátu LaTeX
- `testy` ..... záznamy z testování
  - `IT_Investice_1`
  - `IT_Investice_2`
  - `Nosk_str40`
  - `Nosk_str47`
  - `sec_3_3_1`
  - `sec_3_3_2`
  - `XLS_Scholl_1`
  - `XLS_Scholl_2`
- `text` ..... text práce ve formátu PDF
  - `DP_Trnka_Vaclav_2017.pdf` .....
- `xls` ..... doplňující soubory pro Microsoft Excel
  - `rozpad_hodnot.xlsx` ..... předpis pro výpočet hodnoty IT investice