ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Brzobohatý** Jméno: **Miroslav** Osobní číslo: **465947**

Fakulta/ústav: **Fakulta informačních technologií**

Zadávající katedra/ústav: **Katedra softwarového inženýrství**

Studijní program: **Informatika**

Studijní obor: **Webové a softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Aplikace pro podporu plánování nákladů na informační systém

Název diplomové práce anglicky:

Application for Planning Information System Costs

Pokyny pro vypracování:

Navrhněte a vytvořte nástroj, který umožní zájemcům o využití cloudových služeb vyhodnotit náklady, spojené s jejich využíváním v různých scénářích. Postupujte následujícím způsobem:1) Proveďte revizi znalostí problematiky Reálných opcí, kterou jste se zabýval ve své bakalářské práci. 2) Analyzujte možná rozšíření bakalářské práce z jednoduchého binomického stromu na sadu provázaných binomických stromů, zvažte také využití trinomického stromu místo binomického.3) Navažte na aplikaci, vytvořenou v bakalářské práci, a na základě předchozí analýzy navrhněte její rozšíření, které umožní provádět různé "what-if" scénáře a ty hodnotit z pohledu vynaložených nákladů.4) Návrhy implementujte.5) Vytvořenou aplikaci uživatelsky otestujte na scénářích, reflektujících různé reálné situace z pohledu vývoje trhu.6) Vytvořte metodiku/doporučení pro práci s aplikací, která bude brát ohled na způsoby využívání cloudových služeb (především účtovaných nákladů).

Seznam doporučené literatury:

[1] Miroslav Brzobohatý, HODNOCENÍ FLEXIBILITY PROVOZU APLIKACÍ V PROSTŘEDÍ CLOUD COMPUTINGU, bakalářská práce, ČVUT FEL, Praha, 2019

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Pavel Náplava, Ph.D., katedra softwarového inženýrství FIT

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **09.12.2020** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: _____

Ing. Pavel Náplava, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	doc. RNDr. Ing. Marcel Jiřina, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____ _____

Datum převzetí zadání Podpis studenta

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Aplikace pro podporu plánování nákladů na informační systém

Bc. Miroslav Brzobohatý

Katedra softwarového inženýrství
Vedoucí práce: Ing. Pavel Náplava, Ph.D.

26. června 2021

Poděkování

Tímto bych rád poděkoval za spolupráci a pomoc při psaní diplomové práce vedoucímu práce Ing. Pavlu Náplavovi Ph.D. za to, že mi s vypracováním diplomové práce pomáhal, a vždy tu pro mě byl. Také bych chtěl věnovat zvláštní poděkování rodině a přátelům, kteří tu pro mě byli, a vždy mě podporovali, i v časech nejtěžších.

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.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 26. června 2021 . . . . . .

Odkaz na tuto práci

Brzobohatý, Miroslav. *Aplikace pro podporu plánování nákladů na informační systém.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Cílem této práce bylo se zaměřit na řešení problémů s hodnocením flexibility cloudu z jiného úhlu, a to konkrétně podívat se z větší blízkosti na to, co nám říkají výsledky, které získáme z metody reálných opcích, které jsem se věnoval v bakalářské práci. Základem je teorie popisující základní pojmy a vzorce, které se dále používají. Bude nás zajímat, jak promítnout výsledky reálných opcí do čtenáři srozumitelnější a lépe pochopitelnější formy. Na ní navazuje praktická část, která se zabývá samotným vývojem aplikace, která nám umožňuje lepší pohled na pochopení smyslu hodnocení flexibility cloudu. Nástroj obsahuje relativně široké spektrum funkcí, které jsou popsány dále v práci.

Klíčová slova Cloud computing, On-premise, Binomický model, Trinomický model, Flexibilita, Java, Nástroj, Aplikace, IT investice

Abstract

The aim of this work was to focus on solving problems with evaluating the flexibility of the cloud from a different angle, namely to look more closely at what the results we get from the method of real options, which I addressed in my bachelor's thesis. The basis is a theory describing the basic concepts and formulas that are used further. We will be interested in how to project the results of real options into a more understandable and better understandable form for the reader. It is followed by a practical part, which

deals with the development of the application itself, which allows us a better view of the meaning of evaluating the flexibility of the cloud. The tool contains a relatively wide range of functions, which are described later in the work.

Keywords Cloud computing, On-premise, Binomial model, Trinomial model, Flexibility, Java, Tool, Application, IT investments

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Dnes má společnost spoustu nástrojů pro plánování nákladů na informační systém, dokonce existují nástroje využívající sofistikované vzorce a metody, jako jsou reálné opce. Přestože máme všechny tyto nástroje, tak správně ohodnotit investici do IT není lehká záležitost. V první řadě musíme znát celou řadu údajů, včetně těch, které se nedají prakticky určit, jako je například volatilita, která určuje proměnlivost trhu jako takového. Takovou hodnotu dokáže určit jen expert v dané oblasti, který je schopný definovat, jak moc se trh v dané chvíli pohybuje. To máme hodnoty, které vložíme do nějakého vzorce, nástroje a ten nám vrátí nějakou hodnotu, z které máme poznat, zda je pro nás výhodné si pořizovat vlastní infrastrukturu, nebo je lepší pronájem v cloudu. Co přesně výsledná hodnota, která nám říká, zda je lepší cloud nebo on-premise, znamená, nám už ale nikdo neřekne. V této práci se pokusím přiblížit význam této hodnoty takovým způsobem, aby byla bližší skutečnému světu a znázorním ji na příkladech, které si každý umí představit, jako je například určitá konfigurace, z které je vidět, s čím aktuálně pracujeme.

Teoretické kapitoly se věnují problematice reálných opcí, cloud computingu a mapování na konfigurace z toho pohledu, jak to celé funguje v pozadí, co z čeho vychází, a proč tomu tak je. Taky si představíme pojmy a definice, které budeme potřebovat k pochopení dané problematiky. Podíváme se na využití binomického modelu pro náš problém, a vzápětí si přiblížíme trinomický model, který by nám měl poskytnout přesnější hodnoty oproti binomickému. Po představení základních pojmů, se podíváme na samotnou úvahu o zobrazení hodnot získaných z binomického a trinomického modelů do určitých konfigurací a reálných příkladů ze života.

Praktická část se bude věnovat návrhu celé aplikace a její implementaci. Ukážeme si jednotlivé funkce a vysvětlíme si, co dělají a jak fungují. Následovat bude sekce, kde budeme nástroj testovat na příkladech a ukážeme si, co nám výsledky říkají a k čemu nám jsou. V neposlední řadě se vrátíme na začátek a podíváme se na naplnění cílů práce, které jsme si na začátku stanovili, a zhodnotíme možné cesty dalšího zkoumání a vývoje tohoto nástroje.

Pro toto téma jsem se rozhodl, protože jsem se daným tématem zabýval už v bakalářské práci, a tohle je možnost, jak se na to podívat z jiného úhlu a zjistit, jaké nám

tento úhel pohledu přinese výsledky.

Kapitola 1

Hypotézy

Úplně na začátku, než se pustíme do vysvětlování toho, co je *Cloud computing*, *Reálné opce* a další pojmy, které budeme v rámci práce potřebovat, definujeme několik hypotéz, a na závěr se k nim vrátíme a zhodnotíme, které lze potvrdit, a které nikoliv.

1. H_1 - **Trinomický model poskytuje reálnější (přesnější) informace než model binomický**

   Trinomický model díky možnosti zůstat ve stejné úrovni stromu, a tedy nemuset jít nahoru či dolů, poskytuje přesnější, reálnější výsledky.

2. H_2 - **Rozdíl mezi *cloudem* a *on-premise* se chová stejně jako u *metody reálných opcí* či *simulací***

   Očekávaným chováním je, že rozdíl mezi hodnotami, které nám vyjdou z mapování na reálné konfigurace, bude v základě podobný, i když se může jednat o jinak velké částky, tak ten rozdíl cloudu oproti on-premise zůstává.

3. H_3 - **Výsledná hodnota je srozumitelnější než u reálných opcí, kde nemusí každý vědět, co si pod danou hodnout má představit**

   Předpokládáme, že hodnoty které jsou vázany na konkrétní konfigurace, tedy fyzické komponenty, jsou lépe uchopitelné pro člověka než jen číslo, které se objeví a není ničemu přiřazeno, a není teda zcela jasné, co dané číslo vlastně vůbec znamená.

Kapitola 2

IT infrastruktura a Cloud Computing

V této kapitole si představíme, co je to cloud, a stručně si popíšeme jeho vlastnosti. Text je volně převzat z mé bakalářské práce [9], protože se jedná o stejnou tématiku. Na detailní popis se lze podívat v té bakalářské práci, zde jsou citovány jen ty nejdůležitější části důležité pro tuto práci.

2.1 Definice Cloud Computingu

Přeneseně můžeme říci, že se v určitém slova smyslu jedná o službu, ke které má přístup každý počítač, který má přístup k síti. Služba v tomto případě představuje určitou vrstvu, ve které se skrývá software, hardware, datová uložiště, databáze a aplikace. Zákazník, tedy člověk nebo organizace, která bude tuto službu využívat, si podle vlastního uvážení a potřeb pronajme určitý kus samotného hardwaru, nebo případně kompletní balík pro správu infrastruktury.

Za první obecnou definici Cloud Computingu, lze považovat definici poradenské společnosti Gartner, Inc. *„Cloud Computing is a style of computing where massively scalable IT-enabled capabilities are delivered 'as a service' to external customers using Internet technologies"* [10]. Tato definice ve stručnosti říká, že Cloud Computing je způsob sdílení vypočetní kapacity zákazníkům formou služby napříč síťí. Obdobně, jako společnost Gartner, se i ostatní společnosti pokoušely postavit vlastní definici Cloud Computingu. Důkazem toho, že sestavit jednoznačnou definici se ukázalo být těžším úkolem, než se na první pohled zdálo, je existence desítky dalších definic, které se v mnohém liší, ale všechny mají společných několik základních principů, zbylá část definice se vždy liší podle potřeb a úhlu pohledu autorů definice. Místo jednoznačné definice, na které se společnosti nebyly schopny domluvit, se tak začaly stanovovat společné atributy pro všechny definice, které charakterizují Cloud Computing a zároveň ho jednoznačně odlišují od ostatních technologií.

Jeden z prvních, kdo tyto atributy stanovil, byla opět společnost Gartner, Inc. Samozřejmě i definice jednotlivých atributů charakterizujích Cloud Computing se potkaly s protichůdnými názory, ale i navzdory těmto názorům se tyto definice považují za obecné

výchozí vlastnosti *Cloud Computingu*.

Nakonec bylo definováno pět základních atributů [11]:

1. **Princip služby (Service-Based)**

   Zákazník a poskytovatel jsou navzájem jednoznačně odděleni pomocí přesně definovaného rozhraní, které se nazývá službou. Tato služba staví technickou stránku realizace pro zákazníka do postavení černé skřínky (black box), která ukazuje směrem ven k zákazníkovi pouze rozhraní připravené přijímat požadavky, aniž by bylo nutné se starat o to, jak je nástroj realizovaný uvnitř. Samotný provoz a nastavení této "černé skříňky"je z pohledu zákazníka zcela automatické.

2. **Škálovatelnost a elasticita (Scalable and Elastic)**

   Škálovatelnost dává zákazníkovi možnost upravovat automaticky nebo ručně požadavky na výkon a kapacitu, kterou aktuálně potřebuje a je schopen využít. Elasticita popisuje schopnost sdílených prostředků se přizpůsobovat požadavkům zákazníka, a to jak směrem dolů, tak i směrem nahoru. Dalo by se říci, že se jedná o strukturu, která se automaticky přizpůsobuje podle toho, jak ji zákazník využívá, do ideálního vyrovnaného stavu.

3. **Sdílení (Shared)**

   Základním kamenem celé služby je velmi robustní IT infrastruktura postavená za účelem sdílení a uspokojení potřeb co nejvíce zákazníků. Infrastruktura je sdílená mezi velkým počtem různých zákazníků. S nevyužitými zdroji lze libovolně manipulovat a docílit tak co největší optimalizace výkonu a využití dostupných zdrojů na maximum.

4. **Placení za využití (Metered by Use)**

   Poskytovatel služeb sleduje a měří využití zdrojů zákazníkem a podle definovaných modelů pak vypočítává měsíční sazbu. Vždycky platí, že zákazník platí jen za to, co opravdu využil.

5. **Používání prostřednictvím internetu (Uses Internet Technologies)**

   Ze samotného názvu atributu vyplývá, že jde o to, jak se zákazník ke službě dostane. Zákazník se ke službě dostane prostřednictvím internetu a síťových technologií, jako jsou například protokoly HTTP a HTTPS.

2.2 Výhody a nevýhody Cloud Computingu

Tak jako všechno, tak i Cloud Computing má své výhody, pro které je využíván, a své nevýhody, pro které je kritizován. V následujích dvou podkapitolách si ve stručnosti představíme ty nejvýznamnější z nich.

2.2.1 Výhody Cloud Computingu

1. **Žádné či malé investice**

   Jelikož se celá infrastruktura nachází u poskytovatele, tak není potřeba investovat do vlastní infrastruktury. Maximálně se zde mohou nacházet výdaje na různé licence.

2. **Žádné opakované investice do infrastruktury**

   V budoucnosti nehrozí opakované investice do nových hardwarových komponent, které se většinou každých 3 až 5 let obnovují v závislosti na podpoře výrobce. U Cloud Computingu se tato starost přesouvá na poskytovatele služby. Samostný zákazník se o stav infrastruktury nestará, pro něj se tváří stále ve stejném stavu.

3. **Žádné skryté výdaje**

   Nehrozí, tak jako u *on-premise* infrastruktury, že se náhodně rozbije disk, že bude potřeba reinstalovat server nebo že dojde k jiným situacím, které se náhodně objevují.

4. **Není potřeba správa a údržba**

   Veškerou starost o stav, zabezpečení a hladký běh přebírá poskytovatel Cloud Computingu.

5. **Nezávislost na vlastních zaměstnancích**

   S možností přemístit celou IT-infrastrukturu k někomu jinému souvisí nadbytečnost mít vlastní IT oddělení, které by se o infrastrukturu staralo.

6. **Možnost dynamicky měnit kapacitu**

   Pravděpodobně největší předností Cloud Computingu je možnost dynamicky měnit velikost požadovaného výkonu, tak aby část výkonu nezůstávala nevyužitá. Jako ukázku můžeme vzít obchody, které mají své webové stránky. Je jasné, že v létě bude navštěvovat web méně lidí než v prosinci o svátcích, a tudíž pouze v prosinci potřebuje vyšší výkon než ve zbylých měsících.

7. **Rychlost nasazení**

   Rychlost nasazení oproti *on-premise* infrastruktuře, kde se musí pořídit všechny komponenty a následně je nutné sestavit, je prakticky okamžitá, protože v pronajmuté infrastruktuře je už vše nastavené a může se rovnou začít pracovat.

2.2.2 Nevýhody Cloud Computingu

1. **Někdy vyšší ceny než za realizaci svépomocí**

   Může se stát, že cena za pronajmutou infrastrukturu může být vyšší než pořízení a spravování vlastní infrastruktury, a to hlavně z dlouhodobějšího hlediska u specifických modelů Cloud Computingu, kde jsou přirážky za určité benefity, jako je např. velmi vysoká dostupnost služby.

2. **Zákazníkova data jsou uschována na cizí infrastruktuře**

   I přesto, že data jsou šifrována a na serverech poskytovatele jsou pravděpodobně zákazníkova data lépe zabezpečena, než by byla na vlastních serverech, tak se stále nachází na cizí infrastruktuře, kterou spravuje někdo jiný.

3. **Zákazníkova data putují internetem**

   Nesmíme zapomenout, že data cestují napříč internetem vícekrát než by bylo nezbytně nutné a může se stát, že při větším útoku, i přesto, že jsou zašifrována, se k nim může někdo dostat.

4. **Volba hardware a software je limitována poskytovali Cloud Computingu**

   Není možnost měnit základní konfiguraci a nastavení hardwaru. To znamená, že zákazník nemůže ovlivnit to, jaké aplikace či jaké nastavení komponent by chtěl, ale musí se přizpůsobit nabídce poskytovatele.

5. **Někdy pomalejší reakční doba**

   Jelikož poskytoval Cloud Computingu se může nacházet prakticky kdekoliv na světě a tím pádem se kdekoliv na světě nachází i hardware a software, který nabízí. V důsledku může docházet místy k výpadkům či prodlevám. Je rozdíl, když se připojujete k serveru v České republice z České Republiky nebo z USA.

2.3 Shrnutí

V této kapitole jsme se dozvěděli, co je vlastně *Cloud Computing*, k čemu se používá, jaké poskytuje výhody a nevýhody v oblasti vytváření IT infrastruktury. Tou hlavní výhodou je schopnost se flexibilně přizpůsobovat aktuálním požadavkům a minimalizovat tím nadbytečné výdaje. Dnes, když se vytváří IT infrastruktura, tak se příliš neřeší, jak jí udělat efektivně. Mnozí nejspíš ani nevědí, že je i jiná možnost než *on-premise* IT infrastruktura, a v případě, že vědí, že existuje možnost *Cloud Computingu*, tak nejsou schopní porovnat, jaká varianta je pro ně výhodnější a vybírají na základě jednoduchých nástrojů, které nezohledňují všechny faktory. Nejvíce opomíjeným faktorem je právě flexibilita cloudu. Na tuto problematiku už vznikají práce [12] [7] [8] [13], které se zabývají právě tím, jak zohlednit při výpočtu faktor flexibility.

Kapitola 3

Reálné opce

V této kapitole se podíváme na reálné opce. Po domluvě vedoucím, který je i autorem textu [12] jsem použil části, které budeme potřebovat z jeho popisu reálných opcí, a rozšířil je o některé pojmy, jako je např. trinomický model.

V této kapitole si postupně představíme důležité pojmy, které budeme dále potřebovat. Taky jsou zde rozebrány klíčové parametry a vstupy, použité ve druhé části této diplomové práce.

3.1 Hodnota (cena) opce

Při obchodování s opcemi je nutné vědět, jak vzniká a co vyjadřuje jejich hodnota (ce-na). *„Opce jako právo má svoji hodnotu a ta je ovlivněna parametry, které ji charakterizují. Nevztahuje se přímo k hodnotě podkladového aktiva, ale k právu s ním nakládat podle uzavřené opční smlouvy“* [8]. Parametry, které hodnotu *opce* ovlivňují, se týkají podkladového aktiva z hlediska jeho současné ceny a kolísavosti na trhu. Dalšími parametry jsou podmínky uzavřené opční smlouvy (cena, doba do vypršení, typ a druh opce) a ekonomická situace okolí. Z těchto parametrů odvozujeme vnitřní a časové hodnoty opce. Jejich složením dostáváme celkovou hodnotu opce.

3.2 Vnitřní hodnota opce

Vnitřní hodnota opce je rozdílem mezi *současnou cenou podkladového aktiva* **S** v libovolném časovém okamžiku $\mathbf{t} \leq \mathbf{T}$ a smluvní cenou **X**. Tedy cenou, za kterou můžeme podkladové aktivum koupit či prodat v budoucnu. Můžeme ji definovat násle-dujícím způsobem (vzorec **3.1** a **3.2**):

$$C(S,t) = max(S - X, 0), \textit{pro call opci} \tag{3.1}$$

$$C(S,t) = max(X - S, 0), \textit{pro put opci} \tag{3.2}$$

Vnitřní hodnota opce je rovna nule, pokud je rozdíl *spotové a realizační* ceny pro držitele *opce* nevýhodný (záporný) a *opci* by neuplatnil. V opačném případě je rovna výši zisku bez zahrnutí opční prémie. [14]

3.3 Časová hodnota opce

Vnitřní hodnota opce nezahrnuje do výpočtu *opční prémii*, tedy částku zaplacenou při koupení opce. *Časová hodnota opce* je dána rozdílem mezi *opční prémií* a *vnitřní hodnotou opce*. Vyjadřuje množství času, kolik má opce do *konce doby splatnosti*, a zároveň pravděpodobnost, že *opce* přinese svému majiteli do této doby zisk. *Opci* můžeme přirovnat k rozpadajícímu se investičnímu majetku. Pokud jej vlastníci nerozprodají nebo neuplatní do vypršení, stává se bezcenným.

Vzhledem k tomu, že časová hodnota opce je odrazem vlivu nabídky a poptávky po dané opci na trhu, dalším faktorem, majícím vliv na časovou hodnotu opce, je *volatilita*. O časové hodnotě opce lze tedy *„zjednodušeně říci, že je to částka, kterou je ochoten zaplatit kupující prodávajícímu opce za naději, že se během doby do vypršení opce příznivě změní podmínky na trhu"* [8]. S vyšší *volatilitou* podkladového aktiva roste pravděpodobnost, že se opci vyplatí / nevyplatí uplatnit. Hodnota opce klesá s přibližujícím se koncem doby splatnosti a snižující se šancí na změnu. Nejvyšší časo-vou hodnotu má opce na penězích (**At the Money**), kde jsou si tyto dvě ceny rovny nebo velmi blízké. U této opce je velká pravděpodobnost, že bude brzo v penězích (**In the Money**) a tak se zvyšuje její časová hodnota [8], [1].

3.4 Parametry ovliňující hodnotu (cenu) opce

Klíčové faktory, které ovlivňují hodnotu (cenu) opce byly již z větší části shrnuty v kapitolách *o vnitřní* a *časové hodnotě* opce. Obecně lze konstatovat, že *hodnota opce* je přímo ovlivňována faktory, které se týkají hodnoty podkladového aktiva, podmínek uzavřené opční smlouvy a ekonomické situace okolí. Jedná se především o následující parametry [8], [14], [1]:

1. Současná (spotová, okamžitá, aktuální) cena **S** podkladového aktiva
2. Realizační (strike, smluvní, expirační) cena **X** podkladového aktiva
3. Čas, zbývající do doby vypršení opce (expirační doba, Expiration Date) **T**
4. Bezriziková úroková míra (úroková sazba) **r**
5. Volatilita podkladového aktiva σ (popřípadě $\sigma 2$)
6. Dividendy

Vliv jednotlivých faktorů na *hodnotu opce* zachycuje 3.1.

	Růst parametru ▲		Pokles parametru ▼	
	Kupní opce	Prodejní opce	Kupní opce	Prodejní opce
Současná cena	▲	▲	▼	▲
Realizační cena	▲	▲	▼	▼
Expirační doba	▲	▲	▼	▼
Úroková sazba	▲	▲	▼	▼
Volatilita	▲	▼	▼	▲
Dividendy	▼	▲	▲	▼

Obrázek 3.1: Vliv vstupních faktorů na hodnotu opce [1]

Symbolem $\triangle$ je znázorněn růst hodnoty faktoru a hodnoty opce. Symbolem $\triangledown$ je znázorněn pokles hodnoty faktoru a hodnoty opce.

3.5 Volatilita podkladového aktiva

Volatilita vyjadřuje míru rizika změny ceny *podkladového aktiva* za určité období. Charakterizuje stálost hodnoty a je měřena pomocí směrodatné odchylky σ, popřípadě rozptylu $\sigma 2$. Jedná se o parametr, který se ze všech vstupních údajů měří nejhůře. Obvykle se pro výpočet používají historické kurzy *podkladového aktiva (historická volatilita)* nebo *Black-Scholesův vzorec (implicitní volatilita)* [14]. Využitím tohoto parametru se do výpočtu hodnoty *opce* zahrnuje *flexibilita podkladového aktiva*, která nám jako vstupní hodnota chyběla u běžných metod.

3.6 Binomický model

Zatímco *Black-Scholesův model* předpokládá spojitý proces změny ceny podkladového aktiva, nespojité modely, kam patří i *binomický*, vychází z předpokladu, že celkový čas životnosti opce **T** je možné rozdělit do konečného množství dílčích úseků **n**. Mimo *binomické modely* sem patří také trinomické až n-stupňové modely. Základem výpočtu *binomického modelu* je předpoklad, že v každém časovém bodě **t** může s pravděpodobností **p** dojít k různým stavům světa **n**. Stav růstu s koeficientem **u** a pravděpodobností **p** a poklesu s koeficientem **d** a pravděpodobností **(1-p)**.

Model byl poprvé představen v roce 1979 [15]. Vychází z několika předpokladů [8]:

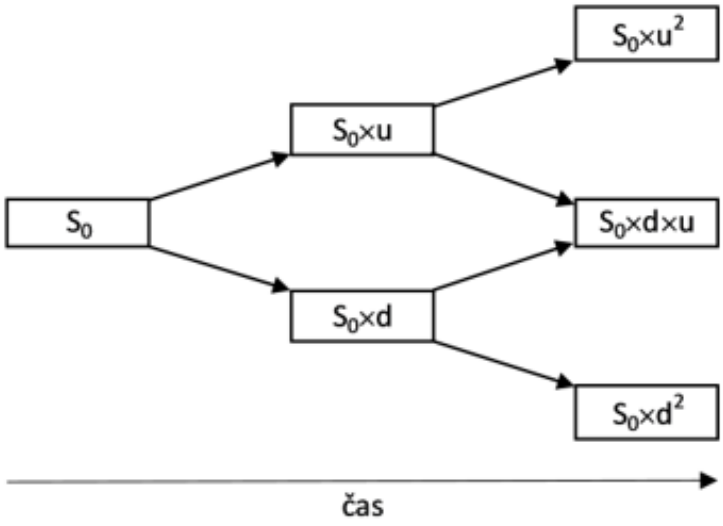
1. Neexistuje možnost arbitráže (nelze dosáhnout bezrizikového zisku)

2. Platí zákon jedné ceny (jestliže mají dvě různá aktiva v budoucnu stejnou hodnotu, pak za předpokladu nemožnosti arbitráže musí mít dnes stejnou cenu)

3. Existují dokonalé trhy (neexistují transakční náklady a daně, neexistuje omezení na krátký prodej, podkladové aktivum je nekonečně dělitelné)

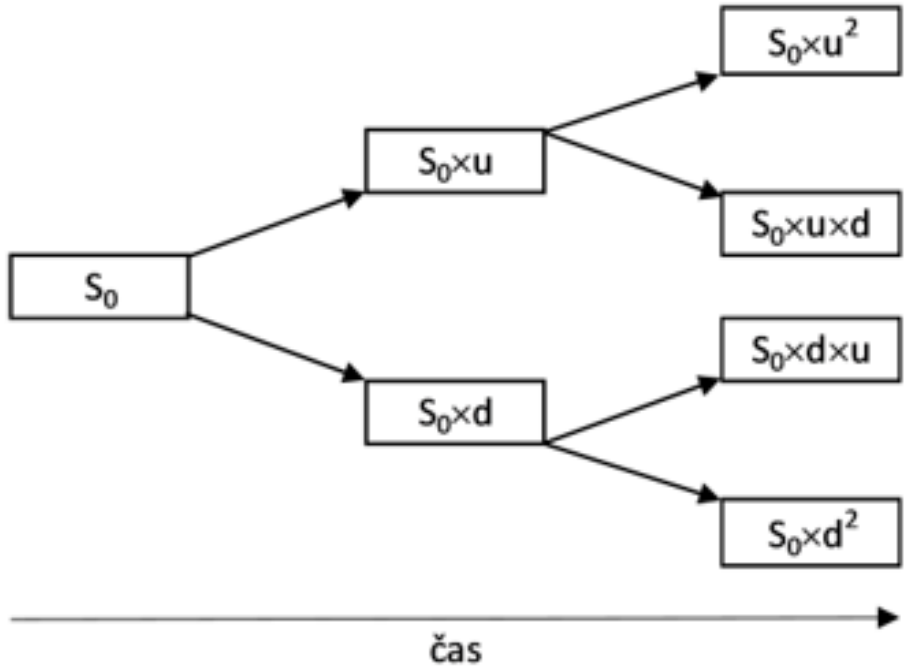
4. Výnos jakéhokoliv aktiva je roven bezrizikové sazbě

Tvorba modelu vychází z předpokladu, že vstupní parametry **u**, **d**, *volatilita podkladového aktiva* a pravděpodobnosti pohybu p a (1-p) jsou konstantní. Ukázku tvorby bi-nomického stromu pro dvě úrovně demonstruje 3.2.



Obrázek 3.2: Ukázka tvorby rekombinačního binomického stromu [2]

Hodnota $\mathbf{S_0}$ reprezentuje počáteční hodnotu aktiva. Standardní proces prvotní tvorby binomického stromu probíhá tak, že pro každý časový úsek **t** se hodnota *podkladového aktiva* buď zvýší o koeficient **u** ($\mathbf{S_n} * \mathbf{u}$) s pravděpodobností **p** nebo sníží o koeficient **d** ($\mathbf{S_n} * \mathbf{d}$) s pravděpodobností **(1 - p)**. Je vidět, že v prvním kroku vzniknou dva uzly ($\mathbf{S_0} * \mathbf{u}, \mathbf{S_0} * \mathbf{d}$), které představují možné hodnoty aktiva na konci prvního období. Na konci druhého období vznikají uzly tři ($\mathbf{S_0} * \mathbf{u^2}, \mathbf{S_0} * \mathbf{u} * \mathbf{d}, \mathbf{S_0} * \mathbf{d_2}$) a stejným způsobem se pokračuje, až do času **T**. Poslední uzly v čase **T** reprezentují rozpětí možných hodnot na konci životnosti **opce**. Rozvoj binomického stromu a počty uzlů v jednotlivých časech **t** lze jednoduše určit například pomocí *Pascalova trojúhelníku* nebo *Binomické věty* [16]. Aby bylo možné správně zkombinovat hodnoty, odpovídající přechodu $\mathbf{u} * \mathbf{d}$ a $\mathbf{d} * \mathbf{u}$, musí platit podmínka, že $\mathbf{u} = \mathbf{1/d}$ [2]. V případě tvorby opčního binomického modelu toto pravidlo platí. Takovýto strom se nazývá rekombinační. Příklad nerekombinační-ho stromu pro dvě úrovně demonstruje 3.3.



Obrázek 3.3: Ukázka nerekombinačního binomického stromu [2]

Koeficient růstu **u** spočítáme pomocí vzorce 3.3 :

$$u = e^{\sqrt{(\sigma^2 * (\frac{T}{N}))}} \tag{3.3}$$

Koeficient poklesu d spočítáme pomocí vzorce 3.4 :

$$d = \frac{1}{u}, d = e^{-\sqrt{(\sigma^2 * (\frac{T}{N}))}} \tag{3.4}$$

Posledními parametry, které jsou zapotřebí pro výpočet hodnoty opce, je pravděpodobnost nárůstu **p**, kterou spočítáme podle vzorce 3.5:

$$p = \frac{(1+r)^{\frac{T}{n}} - d}{u - d} \tag{3.5}$$

a poklesu **(1 − p)**. Se znalostí těchto koeficientů je možné spočítat hodnotu *evropské kupní opce* **C** pro **n** období dle vztahu 3.6:

$$C = \frac{1}{(1+r)^n} \cdot \sum_{i=0}^{n} \frac{n!}{i! \cdot (n-i)} \cdot p^i \cdot (1-p)^{n-i} \cdot max(S \cdot u^i \cdot d^{n-i} - X, 0) \tag{3.6}$$

a hodnotu *evropské prodejní opce* **P** pro **n** období dle vztahu 3.7:

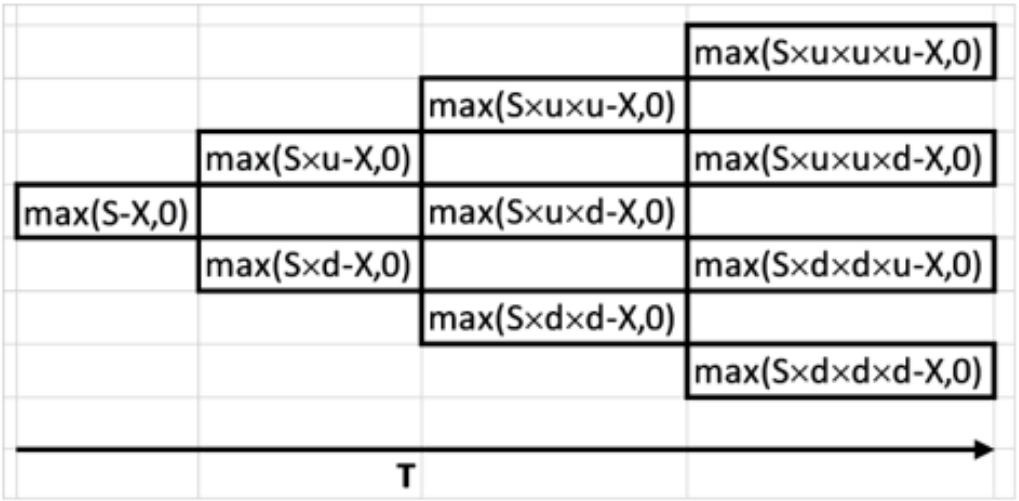
$$C = \frac{1}{(1+r)^n} \cdot \sum_{i=0}^{n} \frac{n!}{i! \cdot (n-i)} \cdot p^i \cdot (1-p)^{n-i} \cdot max(X - S \cdot u^i \cdot d^{n-i}, 0) \tag{3.7}$$

Výpočet *americké opce* je komplikovanější, protože ji lze uplatnit v libovolném okamžiku vygenerovaného binomického stromu. Na rozdíl od *evropské opce* to znamená, že vygenerovaný binomický strom se neprochází jen jednosměrně zleva doprava, ale pro stanovení výsledné hodnoty je nutný druhý krok, ve kterém se strom prochází také zprava doleva. Výpočet je založený na stanovení *vnitřní hodnoty opce* v každém uzlu vygenerovaného uzlu. Pro *evropskou opci* platí, že její vnitřní hodnotu v libovolném uzlu binárního stromu lze spočítat podle vzorce 3.8 v případě *kupní opce* a vzorce 3.9 v případě *prodejní opce*.

$$C_{i,n} = \max\left(S_{i,n} - X, 0\right) \tag{3.8}$$

$$P_{i,n} = \max\left(X - S_{i,n}, 0\right) \tag{3.9}$$

Index **i** znamená i-tou pozici uzlu v období **n**. Hodnota $\mathbf{S_{i,n}}$ vyjadřuje aktuální cenu podkladového aktiva, spočítanou pomocí odpovídají kombinace koeficientů **u** a **d**. Výstupem je vnitřní *hodnota evropské kupní opce* $\mathbf{C_{i,n}}$ nebo *hodnota evropské prodejní opce* $\mathbf{P_{i,n}}$ v uzlu s pozicí **i**, **n** . Pro výslednou *hodnotu evropské opce* je nutné znát jen *vnitřní hodnoty opce* v posledních uzlech, tj. uzlech v čase **T**, které se následně zkombinují a *diskontují* koeficientem **r** (bezriziková úroková sazba) dle vzorce 3.6 nebo 3.7. Výpočet vnitřní hodnoty pro *kupní opci* a tři období znázorňuje 3.4.



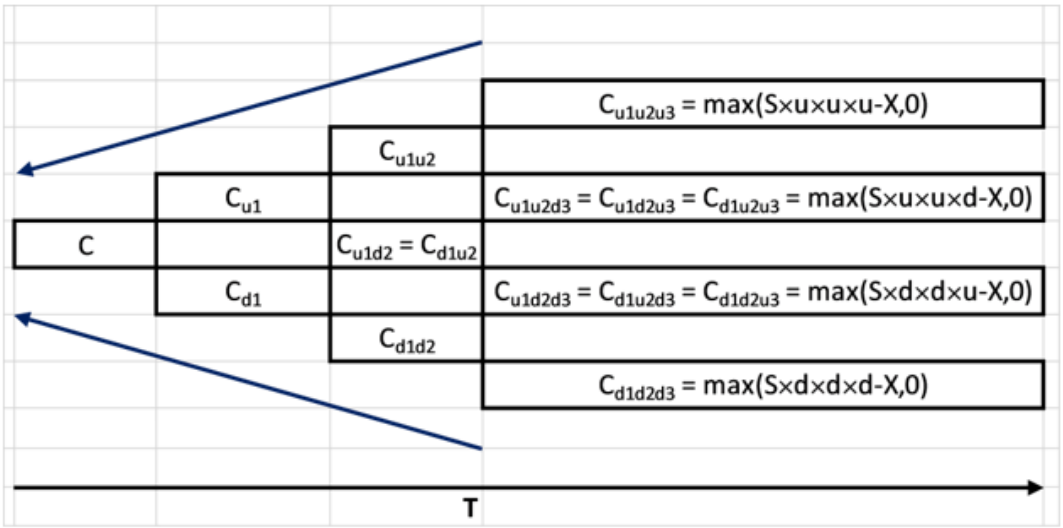
Obrázek 3.4: Výpočet vnitřní hodnoty kupní opce pro tři období [3]

U *americké opce* je výpočet *vnitřní hodnoty* komplikovanější, protože je nutné započítat také hodnoty *opce* pro jednotlivé vnitřní uzly stromu pro případ, že bude opce uplatněná v libovolném z nich. *Vnitřní hodnoty* se proto, mimo poslední úrovně, kde jsou stejné jako u *evropských opcí*, počítají pro *americkou kupní opci* dle vzorce 3.10 a americkou prodejní opci dle vzorce 3.11.

$$C_{i,n-1} = \max\left[\frac{1}{1+r} * (p * C_{i,n} + (1-p) * C_{i+1,n}, \max S_{i,n-1} - X, 0)\right] \tag{3.10}$$

$$P_{i,n-1} = \max\left[\frac{1}{1+r} * (p * C_{i,n} + (1-p) * C_{i+1,n}, \max X - S_{i,n-1}, 0)\right] \tag{3.11}$$

Vzorce 3.10 a 3.11 reprezentují zpětný průchod binomickým stromem zprava doleva a znamenající hodnotu, kterou bychom v daném uzlu získali, kdyby se *podkladové aktivum* prodalo. Vždy se srovnává aktuální hodnota s možnou, *diskontovanou*, budoucí hodnotou. V kořenu stromu získáme výslednou celkovou *hodnotu americké opce*. U *evropské opce* není nutné toto srovnání provádět, protože uplatnit *opci* lze až v čase **T**, takže se vždy hodnotí až poslední úroveň, která se jednorázově *diskontuje*. Ukázku zpětného propočtu hodnoty *americké kupní opce* zobrazuje 3.5.



Obrázek 3.5: Výpočet hodnoty americké kupní opce pro tři období [3]

Mezi výhody binomického modelu patří možnost jeho využití pro výpočet hodnoty jak evropských, tak amerických opcí. Zvyšování množství intervalů **n**, na které je rozdělena doba do vypršení opce, zvyšuje přesnost výpočtu. Model předpokládá, že ve všech obdobích je pravděpodobnost růstu **p** a poklesu **(1-p)** stejná. Nevýhodou je

závislost na správném stanovení indexu poklesu **d** a růstu **u**. Z hlediska porovnání výpočtu *evropské opce* a *americké opce* platí, *že hodnota americké opce* je díky vyšší *flexibilitě* vyšší nebo stejná jako *hodnota opce evropské* [14].

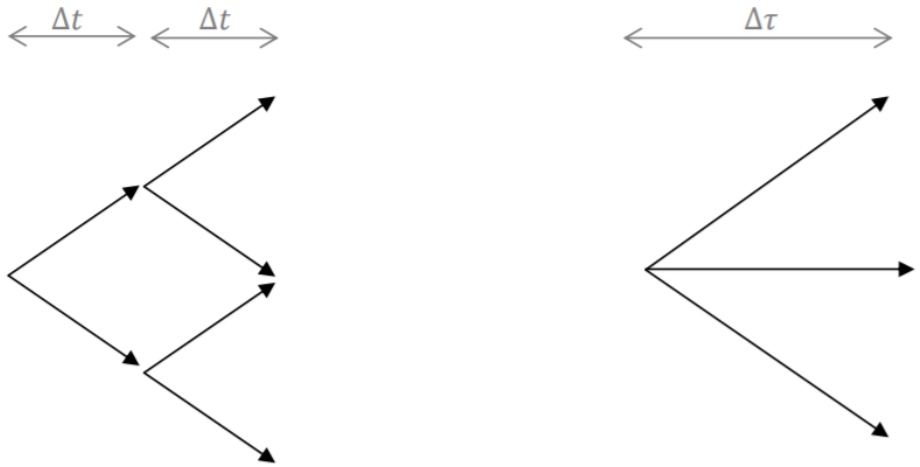
3.7 Trinomický model

Text a informace jsou převzaty z práce [4].

Následující kapitola pojednává o trinomickém modelu oceňování opcí, který zkonstruoval v roce 1986 Phelim Boyle. Trinomický model je podobný předcházejícímu binomickému modelu a lze ho použít k ocenění evropských i amerických opcí.

3.8 Vztah binomického a trinomického modelu

Když porovnáme binomický a trinomický model oceňování opcí, můžeme si všimnout, že spojením dvou kroků o délce Δt z binomického stromu vzniká strom trinomický s jedním krokem délky $\Delta\tau$. Tento fakt je znázorněn na následujícím obrázku 3.6.

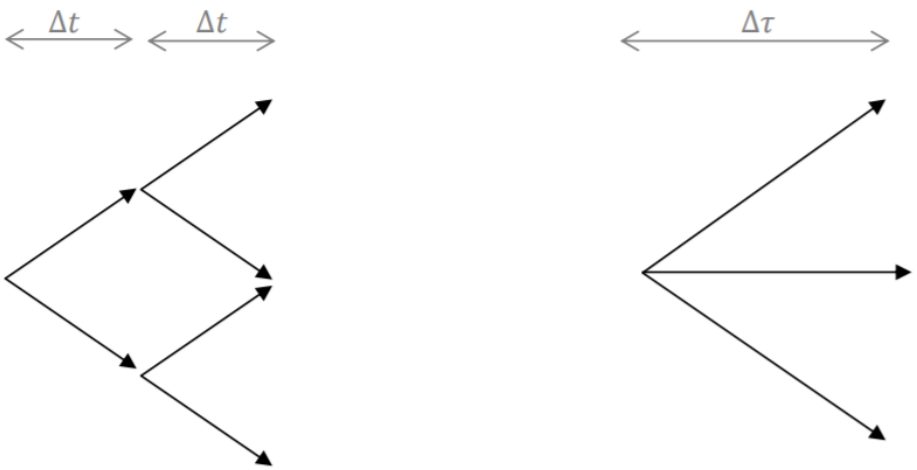


Obrázek 3.6: Vztah binomického a trinomického stromu [4]

3.9 Trinomický model na jedno období

Nejprve představíme ocenění evropské Call opce trinomickým modelem. Předpokládáme, že v každém časovém období můžou nastat tři následující možnosti: s pravděpodobností pu kurz vzroste na uS, s pravděpodobností pd klesne na dS a s pravděpodobností p_m

se tento kurz nezmění a bude mít stále hodnotu S. Tyto možnosti reprezentuje obrázek 3.7.



Obrázek 3.7: Trinomický model pro jedno období [4]

V čase T životnost opce vyprší a její hodnota je stejně jako u binomického modelu rovna její vnitřní hodnotě, tedy

$$C = \max(0, S_t - RC) \tag{3.12}$$

konkrétně

$$C_u = \max(0, uS - RC) \tag{3.13}$$

$$C_m = \max(0, S - RC) \tag{3.14}$$

$$C_d = \max(0, dS - RC) \tag{3.15}$$

Hodnoty proměnných pu, pm, pd, u, d vycházejí z následujících tří podmínek:

1. Součet pravděpodobností setrvání, poklesu a vzrůstu ceny akcie je roven jedné

$$p_u + p_m + p_d = 1 \tag{3.16}$$

2. Aby nemohlo dojít k arbitráži, střední hodnota změny hodnot koncových stavů kurzu podkladové akcie S během jednoho kroku délky $\Delta\tau$ je rovna výnosu s bezrizikovou úrokovou mírou

$$Se^{r\Delta\tau} = p_u uS + p_m S + p_d dS \tag{3.17}$$

po úpravě

$$e^{r\Delta\tau} = p_u u + p_m + p_d d \tag{3.18}$$

3. Poslední podmínka využívá směrodatné odchylky $\sigma\sqrt{2\Delta\tau}$ relativní změny ceny akcie během jednoho kroku. Rozptyl promptního kurzu akcie je roven $S^2\sigma^2 2\Delta\tau$ a na základě definice rozptylu diskrétní náhodné veličiny platí

$$S^2\sigma^2\Delta\tau = p_u u^2 S^2 + p_m S^2 + p_d d^2 S^2 - S^2(p_u u + p_m + p_d d)^2 \tag{3.19}$$

po úpravě

$$\sigma^2\Delta\tau = p_u u^2 + p_m + p_d d^2 - (p_u u + p_m + p_d d)^2 \tag{3.20}$$

Druhá a třetí podmínka vychází stejně jako u binomického modelu z podmínky konvergence k Black-Scholesovu modelu a z pozorování, že dva kroky binomického modelu tvoří jeden krok modelu trinomického. Analogicky k binomickému modelu předpokládáme $d = \frac{1}{u}$ a po vyjádření neznámých z rovnic 3.16, 3.18 a 3.20 získáváme míru vzestupu pro trinomický model ve tvaru

$$u = e^{\sigma\sqrt{2\Delta\tau}} \tag{3.21}$$

Míra poklesu má tvar

$$d = e^{-\sigma\sqrt{2\Delta\tau}} = \frac{1}{u} \tag{3.22}$$

Pravděpodobnost vzrůstu a poklesu v trinomickém modelu vypočítáme jako

$$p_u = \left(\frac{e^{\frac{r\Delta\tau}{2}} - e^{\sigma\sqrt{\frac{\Delta\tau}{2}}}}{e^{\sigma\sqrt{\frac{\Delta\tau}{2}}} - e^{-\sigma\sqrt{\frac{\Delta\tau}{2}}}}\right)^2 \tag{3.23}$$

a

$$p_u = \left(\frac{e^{\sigma\sqrt{\frac{\Delta\tau}{2}}} - e^{\frac{r\Delta\tau}{2}}}{e^{\sigma\sqrt{\frac{\Delta\tau}{2}}} - e^{-\sigma\sqrt{\frac{\Delta\tau}{2}}}}\right)^2 \tag{3.24}$$

Ocenění evropské Call opce pomocí trinomického modelu na jedno období získáme pomocí vzorce

$$C = e^{-r\Delta\tau}(p_u C_u + p_m C_m + p_d C_d) \tag{3.25}$$

3.10 Trinomický model na n období

Nyní stejně jako u binomického modelu rozšíříme trinomický model na více období. Na obrázku 3.8 je zobrazen trinomický strom cen opcí. Jednotlivé uzly stromu jsou pro snadnější orientaci označeny pomocí souřadnic (i, j), kde souřadnice i představuje počet kroků $\Delta\tau$ a souřadnice i představuje v případě kladné hodnoty počet vzestupů ceny opce oproti počátečnímu uzlu, v případě záporné hodnoty se jedná o počet poklesů ceny opce oproti počátečnímu uzlu.

Obrázek 3.8: Trinomický strom cen opcí pro více období [4]

Jak již bylo vidět v předchozí části, oceňování trinomickým model je analogické k binomickému modelu. Označme n počet období trinomického modelu délky $\Delta\tau$. Hodnota opce v čase n je známa a má hodnotu

$$C_{n,j} = \max(0, S_n - RC) \tag{3.26}$$

kde j nabývá hodnot z intervalu $\langle -n, n \rangle$.

Stejně jako u binomického modelu postupujeme zpětně trinomickým stromem a v každém uzlu vypočítáme hodnotu opce podle vzorce

$$C_{n,j} = e^{r\Delta\tau}\left(p_u C_{i+1,j+1} + p_m C_{i+1,j} + p_d C_{i+1,j-1}\right) \tag{3.27}$$

3.11 Reálné opce

Rozvoj používání finančních opcí byl inspirací pro vznik opcí reálných. Finanční svět se ve své podstatě podobá světu jiných odvětví a dle [14] lze *„jako opci chápat téměř jakoukoliv situaci v podniku, která pracuje s rozhodováním na základě podnětů 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 bu-doucích peněžních*

toků souvisejících s realizací nějakého projektu.“ V tomto případě se práva netýkají finančních, ale reálných aktiv, a proto se používá pojem *reálné opce*. O *reálných opcích* se mluví také v případě, že podkladovým aktivem jsou komodity. Na rozdíl od *finančních opcí* není u *reálných opcí* jednoznačně definované vlastnictví, předem neexistuje daná exkluzivita vstupu na trhy či přesunu výroby do nákladově výhodnějších oblastí, popřípadě redukce aktiv konkurence [8]. Až na výjimky (například nákup licencí) reálné opce nejsou institucializovány.

Poprvé je pojem *reálná opce* zmíněn v roce 1977 v článku [17] ve spojitosti s rozšířením, odložením a opuštěním projektu na základě budoucí informace. *Reálné opce* lze chápat jako další metodu hodnocení investic. Na rozdíl od nich ale do výpočtu zahrnuje *flexibilitu*, kterou ostatní metody v sobě přímo nemají. Metoda je modifikací metody NPV a lze ji zjednodušeně vyjádřit vzorcem 3.11.

$$\text{Hodnota projektu} = \text{NPV} + \text{Opční hodnota}$$

Opční hodnota je v tomto vzorci rozšířením hodnoty NPV a zahrnuje v sobě flexibilitu, která představuje právo na pozdější přizpůsobení se aktuální situaci. V této kapitole jsou popsány klíčové rozdíly mezi *finančními opcemi* a Reálnými opcemi a jsou specifikovány pojmy.

3.11.1 Flexibilita, volatilita, nejistoty

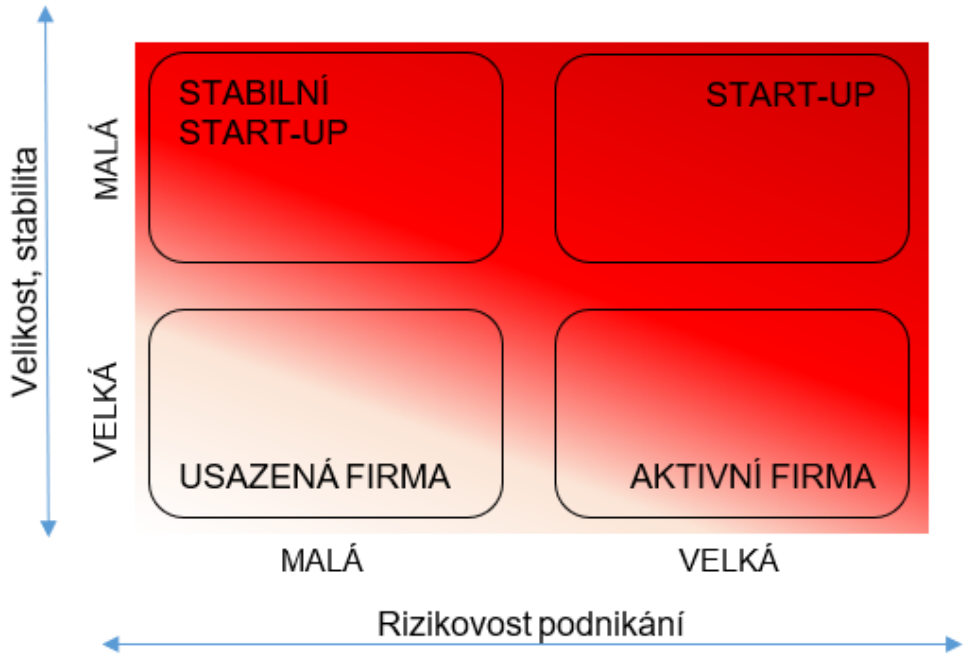
Přidanou hodnotou *metody Reálných opcí* je schopnost zahrnout do výpočtu parametr *flexibility*. Myšlenka *flexibility* neboli pružného reagování a přizpůsobení se trhu, se prolíná rozhodováním ve všech oblastech trhu. Organizace, která dokáže za krátkou dobu odpovídajícím způsobem (*agilně*) reagovat na probíhající změny, má vůči ostatním velkou konkurenční výhodu. „*V souvislosti se strategickým investováním můžeme flexibilitu definovat jako schopnost změnit svá rozhodnutí a v reálném čase investici rozšířit, ukončit, změnit vstupy či výstupy, přemístit apod., a to s cílem maximalizovat svůj užitek z ní*“[8].

Flexibilita je využitelná v situacích, kdy se v průběhu investice změní podmínky provozu nebo podmínky na trhu. Jinými slovy, jde o situace, kdy v době rozhodování o realizaci existovala *nejistota* o stabilitě vstupních podmínek provozu *investice*. Z toho plyne, že ne vždy je nutné *flexibilitu* do výpočtů zahrnovat a že ne vždy zajišťuje větší přidanou hodnotu projektu (konkurenční výhodu). U projektů s vysokou mírou jistoty zahrnutí *flexibility* pouze navyšuje náklady na realizaci bez většího užitku.

Nejistota je obecně používaný termín pro popis něčeho, co neznáme. Buď proto, že nevíme, kam směřuje vývoj, není možné danou věc přesně zjistit nebo je zjistitelná až později. Pro číselné vyjádření nejistoty se požívá termín *volatilita*. Pochází z latinského slova *volare* („létat“) a v překladu znamená kolísavost, kolísání, nestálost, respektive výši a frekvenci změn ceny nebo hodnoty. Vyjadřuje se pomocí směrodatné odchylky nebo rozptylu [8].

Čím je vyšší *volatilita*, tím vyšší je rozptyl možné úspěšnosti projektu. A to jak směrem k vyšším ziskům, tak i vyšším ztrátám. Schopnost přesně odhadnout budoucí

změny okolních podmínek, ovlivňujících projekt, je v tomto případě nižší, protože existuje vysoká míra *nejistoty*, která zvyšuje *riziko* chybného odhadu. Důvodem je ne-stálost v daném segmentu trhu nebo neznalost segmentu, například z důvodu jeho "novosti" . Nízká volatilita naopak představuje stabilnější prostředí. Vychází z dobré znalosti již existujícího segmentu trhu, existující konkurence a rigidnosti s ohledem na možné a smysluplné inovace daného segmentu. Tento trh je dobře předvídatelný a možný rozptyl zisku / ztrát je malý [5]. Podobně lze přistoupit k míře rizikovosti i z pohledu samotné organizace, která se může nacházet v různém okamžiku svého životního cyklu, jak ukazuje 3.9.



Obrázek 3.9: Závislost volatility na životním cyklu organizace [5]

Míra rizikovosti (hodnota *volatility*) je zachycena sytostí červené barvy a čím je barva sytější, tím je její hodnota vyšší. Největší rizikovost je u začínajících, neusazených *start-up* organizací, které se obvykle pohybují na pro ně neznámém, těžko uchopitelném segmentu trhu. Dle [18] lze charakterizovat *start-up* jako firmu s krátkou existencí, malým týmem, vysokým podnikatelským rizikem a s vysokým potenciálem návratnosti v případě úspěchu. Nejnižší míra rizikovosti je naopak u velkých ustálených firem, které výrazným způsobem (velikostí) pokrývají nebo dokonce ovládají stabilní trh například s *utilitami* nebo *komoditami*, u kterého je riziko a pravděpodobnost, že dojde k neočekávané změně velmi nízká. Dle [19] je vysoká míra rizikovosti také u organizací, které jsou velké, usazené, ale mají tendenci vstupovat do nových segmentů trhu a vkládat kapitál do projektů, které jsou velmi často realizovány novými *start-up* organizacemi. Jak zisk, tak i ztráta z těchto investic mohou být velké.

3.12 Zhodnocení Reálných opcí

Ve stručnosti jsme si představili *reálné opce* a vysvětlili nejdůležitější pojmy pro pochopení této problematiky. Dále jsme teorii rozšířili o trinomický model a jeho vztah s binomickým modelem. Což je zároveň pro nás i nejdůležitější část, kterou budeme dále v této práci potřebovat, protože z metody reálných opcí použijeme pouze binomický a trinomický model, dalé se od původní metody odkloníme.

Kapitola 4

On-premise vs Cloud Computing

V této kapitole bude vysvětlena a popsána teorie k tomu, jak spojit hodnoty získané z binomického nebo trinomického modelu na konkrétní konfiguraci. Bude popsáno, na jaké úskalí je třeba si dát pozor a jak je vyřešit. A v neposlední řadě si ukážeme dostupný konfigurátor, který byl použit jako zdroj konfigurací pro tuto práci.

4.1 Mapování konfigurace na Cloud a On-premise

Zde se podíváme nejdříve na mapování Cloudu a následně na mapování On-premise. U obou možností existují určitá úskalí, na která je třeba si dát pozor, a myslet na to, že nemůžeme jenom jednoduše propojit hodnotu s konfigurací.

Níže si ukážeme jeden příklad, aby bylo lépe vidět, o co nám jde.

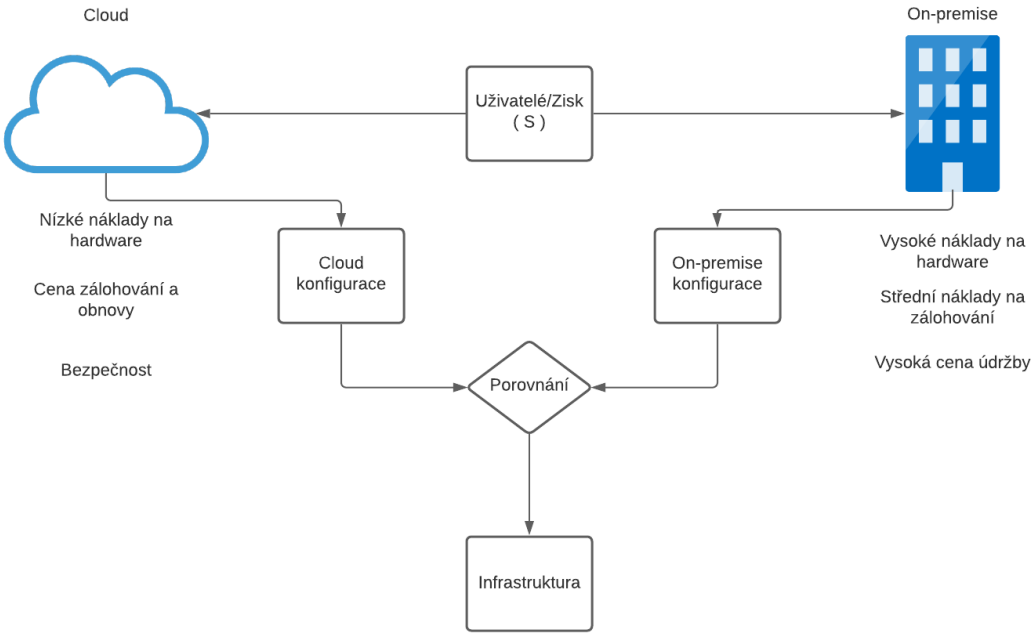
Firma ma určitý počet uživatelů, kteří využívají jejích služeb. Tito uživatelé generují firmě zisk **S** např. 300 000,- Kč. A otázka zní, zda je pro firmu výhodnější zařídit infrastrukturu v cloudu nebo si pořídit celou infrastrukturu on-premise. První otázka je, zda **S** je dostatečně vysoké, aby pokrylo veškeré požadavky a ještě se to firmě vyplatilo. V případě, že to nepokryje všechny náklady, tak je třeba **S** navýšit zdražením služeb.

V případě, že **S** je dostatečně vysoké, tak bude firmu zajímat, zda je výhodnější řešení v cloudu nebo on-premise. To už se snadno zjistí použitím tohoto nástroje, který ukáže kolik které řešení stojí a zároveň ukáže odpovídající konfiguraci k dané částce. Podle toho už lze snadno rozhodnout, které řešení je lepší.

4.1.1 Mapování na Cloud

Spojit hodnotu z vytvořeného stromu s konfigurací je relativně snadné. Víme, že máme zisk, a víme, že daná konfigurace v Cloudu stojí určitou částku, naším cílem je najít ideálně nejlepší možnou konfiguraci, která se vejde do dané částky. Případně by se dalo definovat kolik chceme, aby nám z toho zisku zbylo, abychom i něco vydělali, a nedali všechno na údržbu a provoz infrastruktury.

Takže jediným problémem, který před sebou máme je vytvořit konfiguraci odpovídající dané částce. Pro tuto práci bylo nakonec rozhodnuto využít zjednodušený způsob, kdy



Obrázek 4.1: Cloud vs On-premise [vlastní]

aplikace nebude automaticky vytvářet tyto konfigurace, to by totiž bylo nad rámec této práce. Byl tedy použit způsob, kde máme seznam konfigurací, do kterého byly jednotlivé konfigurace ručně přidány a program bude pracovat pouze z těmito konfiguracemi, když se bude rozhodovat, kterou vybrat.

Pro samotné rozhodování slouží jednoduchý algoritmus, který projede seznam těchto konfigurací a u každé si spočítá, jaký je rozdíl mezi požadovanými náklady a zadanou částkou, pokud rozdíl není zaporný nebo větší než aktuálně uložený, tak si ho uloží, takto nám na konci zůstane konfigurace, která je nejblíže dané hodnotě.

Další úskalí je případ, kdy je zisk tak malý, že se nenajde žádná vhodná konfigurace. V tu chvíli musíme brát v potaz, že nelze uvažovat nulové náklady, protože nějaké minimální požadavky na infrastrukturu vždycky budou, aby mohla firma dál pokračovat v činnosti, tudíž v tu chvíli si pamatujeme poslední konfiguraci a tu použijeme, a jelikož nemáme v toto období podle stromu dostatečný zisk, tak to znamená, že prodělváme.

4.1.2 Mapování na On-premise

V druhé části kapitoly je popsáno mapování hodnot z vytvořeného stromu na on-premise konfigurace. Stojíme před obdobným problémem jako v první části kapitoly u Cloudu. V tomto případě ale musíme zvolit trochu jiný pohled, jak bylo řečeno v práci [12], protože u on-premise je ten problém, že občas musíme jednorázově více investovat do

navýšení výkonu a následně platíme jen provozní náklady, které jsou oproti pořizovací ceně minimální. V takovém případě nelze srovnávat cloud s on-premise, protože jsou tyto varianty nesrovnatelné.

Abychom tento problém vyřešily, musíme stejně jako u metody reálných opcí převést on-premise na něco, co lze srovnávat s náklady na cloud, tedy rozložit prvotní investici do měsíčních poplatků. Pro tento účel zvolíme rozpad na základě výpočtu hodnoty anuity jako v práci [12].

Zjednodušeně lze popsat výpočet celkových nákladů na on-premise infrastrukturu A vzorcem:

$$A = A_{INV} + \sum_{i=1}^{n} A_{OPERi} \tag{4.1}$$

A_{INV} představuje celkovou prvotní investici do pořízení a A_{OPERi} provozní náklady v období i.

To by však způsobilo, že by měsíční náklady na cloud a on-premise nebyly porovnávatelné, protože v období pořízení by náklady byly nesrovnatelně vyšší a to by eliminovalo dopady flexibility. Řešením je právě rozložení investice do všech časových obdobích. Proto tento účel využijeme časový rozpad na základě výpočtu hodnoty anuity. Pro výpočet se běžně používá vzorec:

$$A_i = \frac{rA}{1 - (1 + r)^{-n}} \tag{4.2}$$

Hodnota A zde představuje celkovou jednorázou investici do pořízení. Parametr **r** je bezriziková úroková míra. Jelikož chceme používat období o délce jednoho měsíce, tak musíme použít modifikovaný vzorec:

$$A_i = \frac{A(e^{r_e \Delta t} - 1)}{1 - e^{-r_e \Delta t n}} \tag{4.3}$$

Vzorec je založený na spojitém úročení a využívá efektivní úrokové sazby r_e.

Když investici na pořízené dané konfigurace rozložíme podle tohoto vzorce v našem případě na 12 období, tak už porovnáváme porovnatelné a můžeme už přiřadit správnou konfiguraci dané hodnotě.

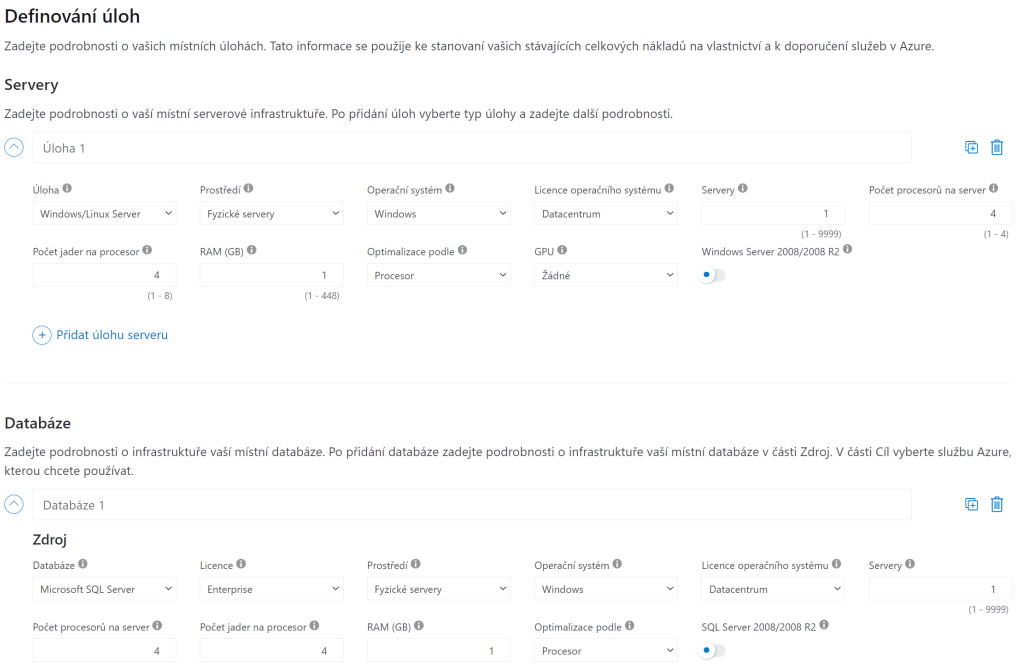
A posledním problémem, který musíme vyřešit je situace, kdy povýšíme on-premise v půlce životnosti projektu a náklady rozložíme do 12 měsíců, tak splácení vychází za dobu životnosti projektu. Abychom tedy mohli správně srovnávat on-premise s cloudem, musíme provést korekci, protože se jedná o nevyužitý výpočetní výkon, který jsme zaplatili, a tyto splátky přičíst zpátky k nákladům na on-premise, aby nebyl oproti cloudu zvýhodněn.

4.2 Dostupné konfigurátory

Existuje více konfigurátorů, které lze použít pro sestavení konfigurace. Tyto konfigurátory nabízejí velké společnosti jako je Amazon, Microsoft a další. Zde si ukážeme jeden z

možných konfigurátorů třetí strany a to konkrétně konfigurátor Microsoft Azure. Vybrán byl na základě toho, že umožnuje nejvíce detailní nastavení konfigurace z nabízených konfigurátorů. Bere například v potaz i cenu elektrické energie, umístění hardwaru a podobně.

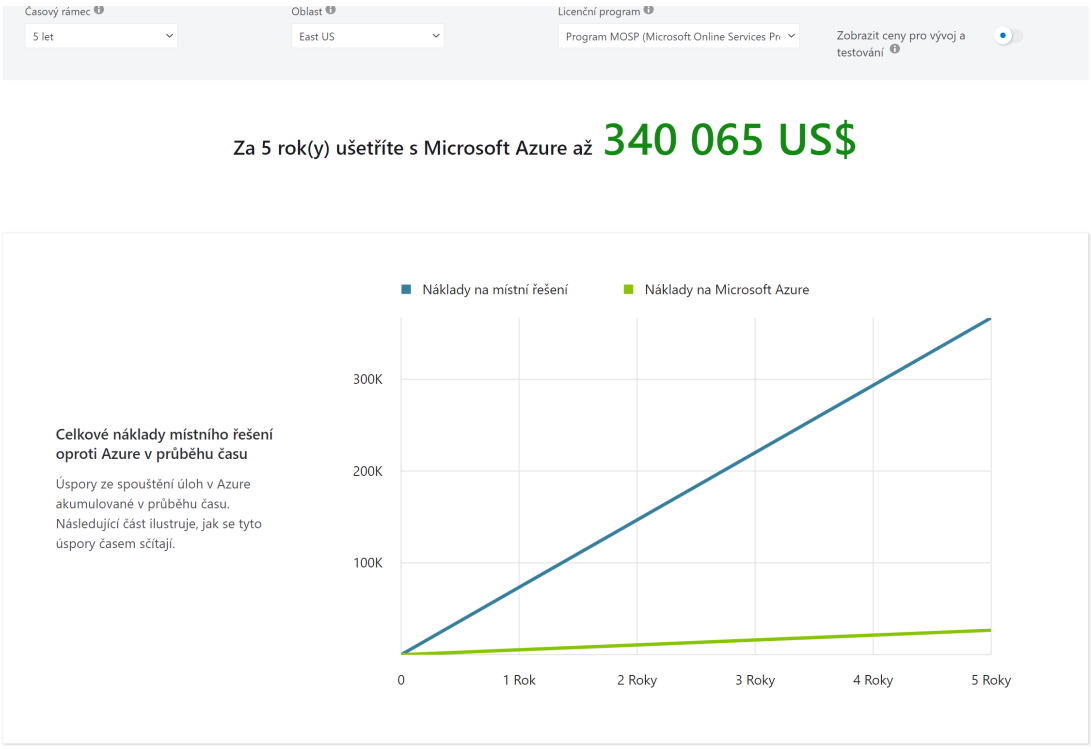
Na obrázku 4.2 můžeme vidět část první stránky vstupních parametrů, které kalkulátor vyžaduje, aby mohl vypočítat, kolik se se službou Microsoft Azure ušetří. Nastavení je celkem dosti podrobné a bere v potaz např. i cenu elektrické energie.



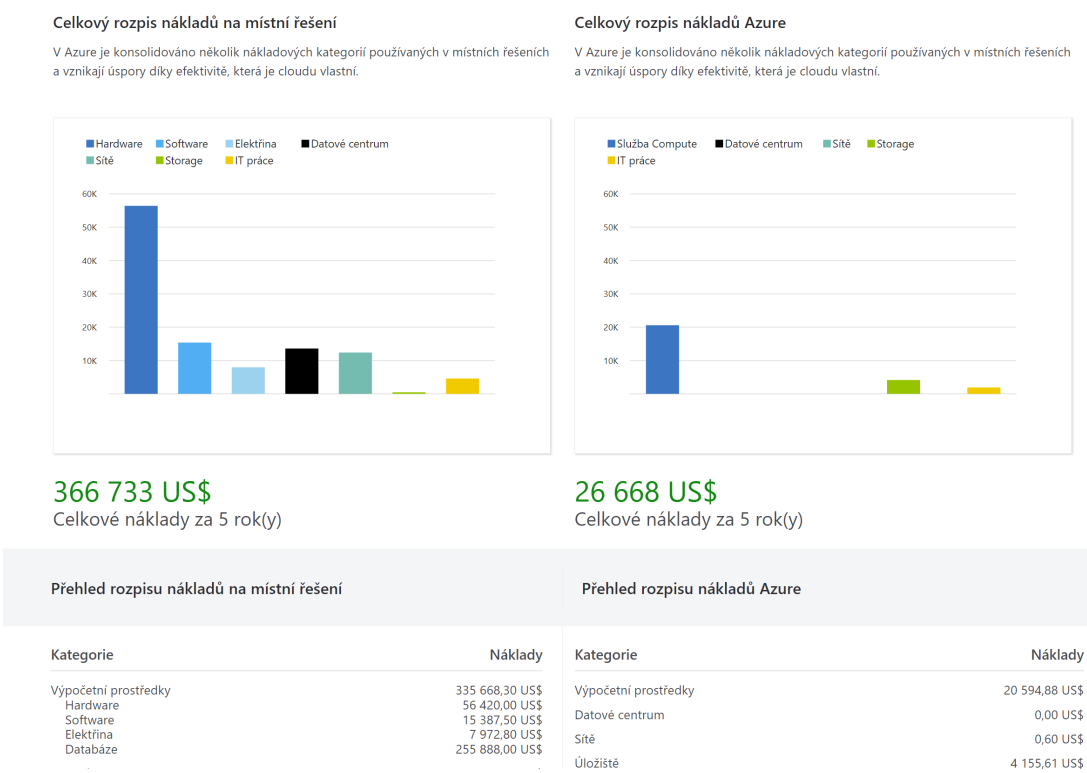
Obrázek 4.2: Zadávání vstupních parametrů [6]

Na dalším obrázku 4.3 je zobrazena část výstupu, který nám kalkulátor vrátí. Jedná se o graf znázorňující křivku nákladů v cloudu a on-premise.

Na posledním obrázku 4.4 lze vidět druhou část výstupu, kde je podrobně rozepsáno, kolik peněz bylo vynaloženo na jednotlivé kategorie. Pod obrázkem je uvedena cena čistě za on-premise řešení a zvlášť cena za řešení v cloudu na Microsoft Azure. Na základě toho lze sestavit konfigurace, které pak lze přidat do nástroje.



Obrázek 4.3: Výsledný graf [6]



Obrázek 4.4: Detailní rozpis nákladů [6]

Kapitola 5

Analýza a návrh řešení

V následující kapitole podrobně rozebereme cíle této práce a podíváme se, jakým způsobem jich bude dosaženo. V první řadě si projdeme souhrn požadavků, které jsme během zpracování práce definovali, a postupně si je popíšeme. Následně si projdeme návrh samotného implementovaného nástroje, který je hlavním přínosem této práce. Ten se bude skládat z popisu datové struktury, kterou použijeme pro ukládání dat, grafického rozhraní aplikace a vnitřní logiky aplikace. Na závěr kapitoly si shrneme technologie použité při vývoji.

5.1 Cíle praktické části práce

Cílem práce bylo na základě znalostí navrhnout a implementovat podpůrný nástroj pro podporu plánování nákladů na informační systém. K tomu využijeme znalostí z předešlých kapitol, kde jsme si představili problematiku Reálných opcí a Cloud computingu. Jedná se o experimentální aplikaci, která může být rozvíjena, pokud se ukáže, že výsledky, které poskytuje, budou užitečné pro další zpracování. Může se taky stát, že zjistíme, že nám dá neurčité výsledky, kterou nebudou použitelné pro vyvozování dalších závěrů. Předkládaná práce staví na výsledcích dalších prací [12] [7] [20], ze kterých byly převzaty základní myšlenky a na nich byla postavena nová aplikace, která se snaží celou problematiku přiblížit z jiného úhlu, lépe pochopitelného pro osoby, které se v dané problematice tolik nepohybují. Hlavním cílem bylo vyvinout nástroj, který by umožnil zobrazit si výsledky binomického či trinomického rozvoje na reálných situacích, a tedy lépe vidět, co dané hodnoty znamenají, a jak ovlivňují výsledek. Zároveň může vytvovořená aplikace sloužit jako určitá alternativa k výpočtu hodnoty IT investice.

5.2 Analýza požadavků

Teď se blíže podíváme na jednotlivé požadavky, které jsme si specifikovali při definování práce a jejího obsahu. Rozděleny budou klasicky na funkčí a nefunkční požadavky, tedy na požadavky na funkcionalitu aplikace a požadavky na výkon a použitelnost.

5.2.1 Funkční požadavky

Funkční požadavky vycházejí ze samotných základů aplikace a byly definovány už na začátku.

1. **Binomický model** - V základu jsme pro jednoduchost nejdříve využili binomický model, který je zároveň i jeden z hlavních požadavků. Je to nosný základ, na kterém celá myšlenka stojí, protože se opíráme o rozvoj v těchto modelech, který dále rozvíjíme a nasazujeme na něj další logiku
2. **Trinomický model** - Dále bylo požadováno zapojení i trinomického modelu oceňování opcí, který by nám měl vracet přesnější hodnoty vzhledem k tomu, že nám nabízí tu možnost zůstat ve stejné úrovni
3. **Navazování stromů** - Možnost v každém koncovém bodě jednoho stromu vytvořit další strom, a takto jich lze navázat libovolné množství s vlastními parametry
4. **Konfigurace** - Aplikace musí umět vzít výsledky z rozvoje binomického / trinomického modelu a vytvořit k nim odpovídající konfiguraci, která bude odpovídat dané hodnotě vycházející z rozvoje binomického či trinomického modelu
5. **Generování / Výběr cest** - Důležitým požadavkem bylo automatické generování optimistické / pesimistické cesty a ještě důležitější možnost si ručně cestu zvolit
6. **Import / Export** - Požadavkem byla možnost načítat a ukládat rozdělanou práci, aby se k uživatel mohl později vrátit
7. **Porovnání On-premise vs Cloud** - Pro lepší představu a vizualizaci rozdílu, je zde možnost porovnat výsledky obou variant
8. **Metoda reálných opcí, Simulace** - Bylo rozhodnutu, že součástí aplikace bude i nástroj navržený a implementovaný v práci [9]

5.2.2 Nefunkční požadavky

Nefunkční požadavky vychází z toho, že vyžadujeme určitou výkonnost a rychlost výpočtu, aby byla aplikace využitelná pro normální používání. Taktéž záleží na designu, aby se aplikace dobře ovládala.

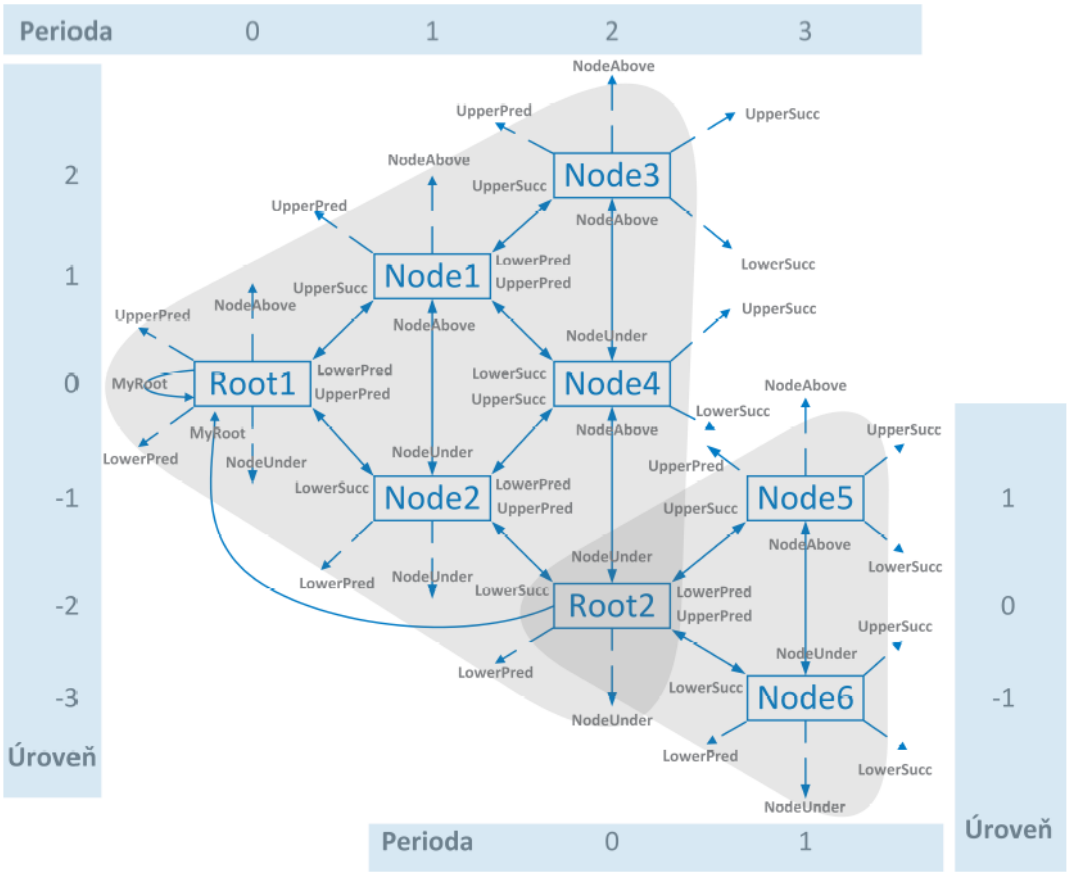
1. **Desktopová aplikace** - Bylo rozhodnuto, že se bude jednat u desktopovou aplikaci

5.3 Návrh

Část návrhu - datová stuktura pro binomický model je stejná jako v mé bakalářské práci [9].

5.3.1 Datová struktura pro binomický model

V první řadě bylo důležité najít datovou strukturu, která by dokázala uchovávat parametry a důležité mezivýpočty pro správné fungování aplikace. U použité struktury bylo důležité, aby dokázala během výpočtu uchovávat data o průchodnosti uzlů a další informace potřebné pro generování stanovených reportů a zároveň aby byla použitelná pro implementaci vedlejší metody reálných opcí. Po krátké úvaze byla vybrána struktura, kterou navrhl a použil Václav Trnka ve své diplomové práci [7]. Jedná se o orientovaný graf, ve kterém si jednotlivé uzly uchovávají reference na všechny uzly v nejbližším okolí. Datová struktura je vidět na obrázku 5.2. Tato struktura je nejen vhodná pro klasický výpočet metody reálných opcí, ale dá se zároveň použít jako nosič informací, jako je průchodnost uzlů při simulacích, ke kterým jinak samotná datová struktura není příliš potřeba.



Obrázek 5.1: Datová struktura pro binomický model [7]

Každý uzel, jak je z obrázku vidět, má celkově šest referencí na okolní uzly.

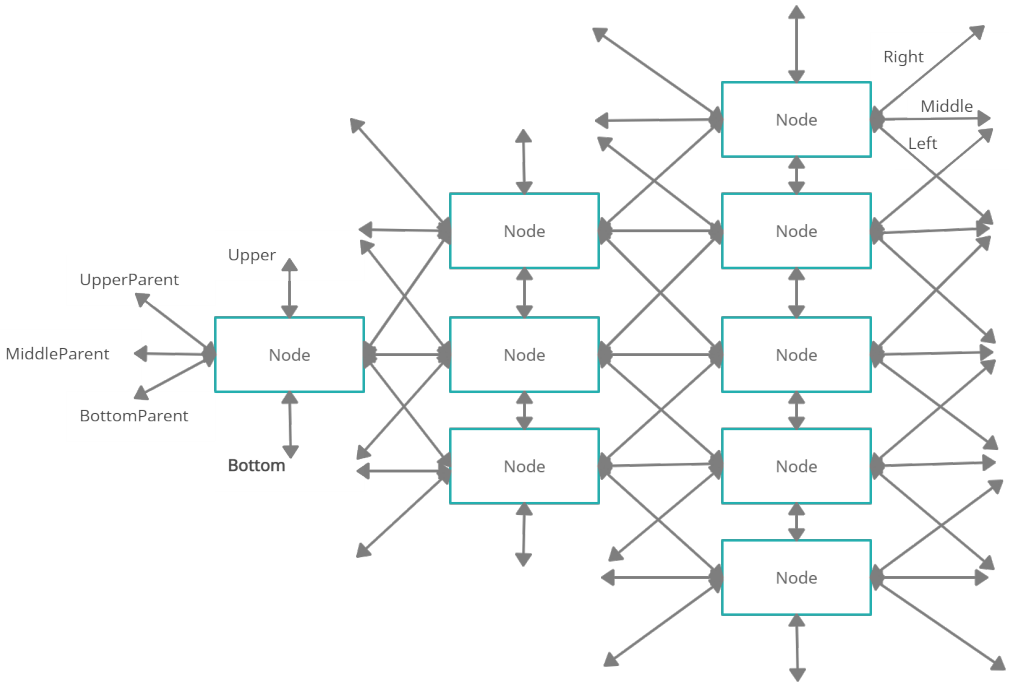
1. **UpperParent** slouží jako označení pro referenci na uzel, který je horním předchůdcem

aktuálního uzlu. Reference je používána především pro schopnost přenášet seznam anuitních splátek.

2. **BottomParent** označuje referenci na uzel, který je spodním předchůdcem aktuálního uzlu. Slouží ke stejnému účelu jako reference na horního předchůdce.
3. **Right** označuje referenci na uzel, který je horním následníkem aktuálního uzlu.
4. **Left** označuje referenci na uzel, který je spodním následníkem aktuálního uzlu.
5. **Upper** je reference na uzel, který se nachází přímo nad aktuálním uzlem a slouží k přechodům mezi jednotlivými úrovněmi.
6. **Bottom** je reference, která ukazuje na uzel, který se nachází přímo pod aktuálním uzlem, a je tedy opakem reference Upper.

5.3.2 Datová struktura pro trinomický model

Datová struktura pro druhý typ modelu - tedy trinomický, vyžaduje úpravu předchozí struktury o přidání několika dalších uzlů a referencí na potomky a rodiče, kteří se u trinomického modelu přidali navíc.



Obrázek 5.2: Datová struktura pro trinomický model [vlastní]

Každý uzel, jak je z obrázku vidět, má celkově osm referencí na okolní uzly.

1. **UpperParent** slouží jako označení pro referenci na uzel, který je horním předchůdcem aktuálního uzlu. Reference je používána především pro schopnost přenášet seznam anuitních splátek.
2. **BottomParent** označuje referenci na uzel, který je spodním předchůdcem aktuálního uzlu. Slouží ke stejnému účelu jako reference na horního předchůdce.
3. **MiddleParent** označuje referenci na uzel, který je prostředním předchůdcem aktuálního uzlu. Slouží ke stejnému účelu jako reference na horního předchůdce.
4. **Right** označuje referenci na uzel, který je horním následníkem aktuálního uzlu.
5. **Left** označuje referenci na uzel, který je spodním následníkem aktuálního uzlu.
6. **Middle** označuje referenci na uzel, který je prostředním následníkem aktuálního uzlu.
7. **Upper** je reference na uzel, který se nachází přímo nad aktuálním uzlem a slouží k přechodům mezi jednotlivými úrovněmi.
8. **Bottom** je reference, která ukazuje na uzel, který se nachází přímo pod aktuálním uzlem, a je tedy opakem reference Upper.

5.3.3 Vytváření stromů

Abychom se mohli posunout k dalším úvahám a výpočtům, tak v první řadě potřebujeme vytvořit základ, kterým je binomický nebo trinomický model. K tomu potřebujeme znát parametry potřebné pro vytvoření těchto stromů:

1. Aktuální cena podkladového aktiva **S**
2. Doba životnosti projektu **T**
3. Míra volatility σ^2
4. Bezriziková úroková míra **r**
5. Počet období **n**
6. Stabilita **c** - tato hodnota udává, když generujeme navazující stromy, tak po jakém počtu období má začít nový podstrom se stejnou nebo jinou volatilitou.

Z těchto parametrů získáme hodnoty u a d, pomocí kterých se už dá vygenerovat vývoj spotové ceny.

Vývoj spotové ceny pak probíhá od kořene stromu a posouvá se postupně dolů. Dle vzorců binomického modelu bude spotová hodnota levého následníka $S * d$ a pravého následníla $S * u$. Pak se rekurzivně to samé provede pro všechny následovníky. Výpočet jako takový skončí, když dojde do listu stromu, tedy krok se bude rovnat počtu období.

S takto vygenerovaným stromem, ve kterém jsou nastaveny všechny potřebné hodnoty, se můžeme posunout dále ke generování / výběru cesty stromem a následně k mapování těchto cest na reálné konfigurace.

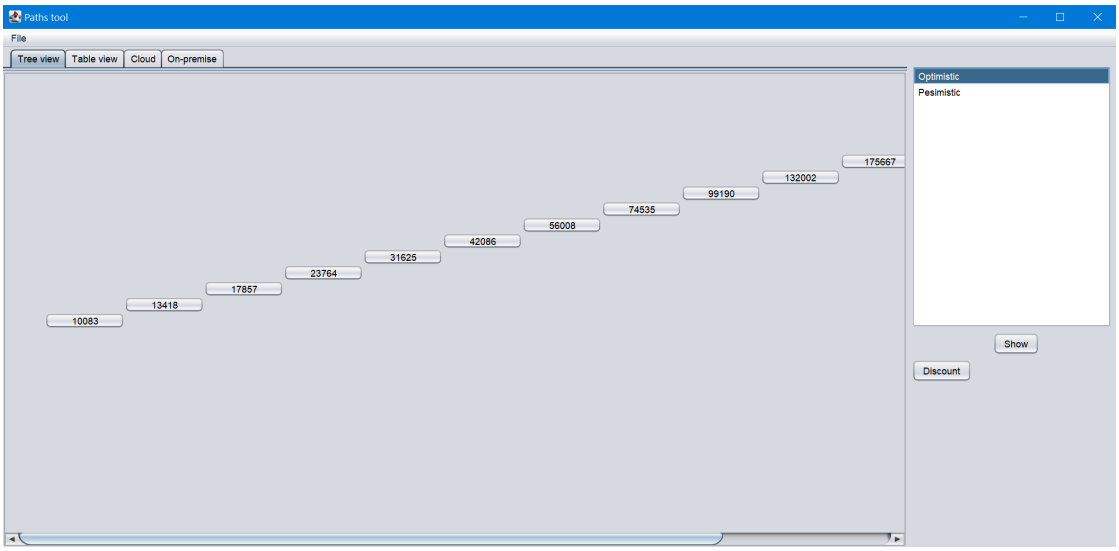
5.3.4 Cesty / Průchody stromem

Po vytvoření požadovaného typu stromu si můžeme začít hrát s tím, jaké průchody nás zajímají, nebo jaký vývoj situace očekáváme. Na základě toho si můžeme vytvořit tolik cest, kolik budeme chtít. Po vybrání a uložení námi požadovaných cest se můžeme v další části aplikace podívat z blízka na jednotlivé cesty a zjistit, co nám říkají.

Pro výběr a ukládání takových cest bylo potřeba navrhnout vhodný algoritmus, který by komunikoval s UI a na základě postupného vybírání cesty eliminoval možnosti, které nemůžou být v tu chvíli už vybrány, protože stromem lze jíti jen dopředu, nelze se vracet zpátky. Pro tuto potřebu má každý node atribut *enabled*, který se nastavuje na false, pokud je node v zóně, kam už z logického hlediska nemůže pokračovat. Toho docílíme tím, že se vrátíme do rodiče aktuálního nodu, a v případě binomického modelu projdeme až do konce druhou větev a všude nastavíme *enabled* na false, podobně u trinomického modelu uděláme to samé pro zbývající dvě možné cesty, s tím že v případě prostřední cesty jdeme jednou rovně a pak směrem od vybrané cesty.

Ve chvíli, kdy dojdeme tímto postupem až na samotný konec stromu, tak při vybrání posledního nodu v cestě se zavolá metoda, která danou cestu uloženou v podobě seznamu nodů, přes které vede, uloží do sdíleného objektu, který si jednotlivá okna aplikace posílají mezi sebou, aby mohla pracovat nezásvisle na sobě se stejnými daty.

Jak taková cesta vypadá je vidět na obrázku 5.3.



Obrázek 5.3: Grafické znázornění cesty

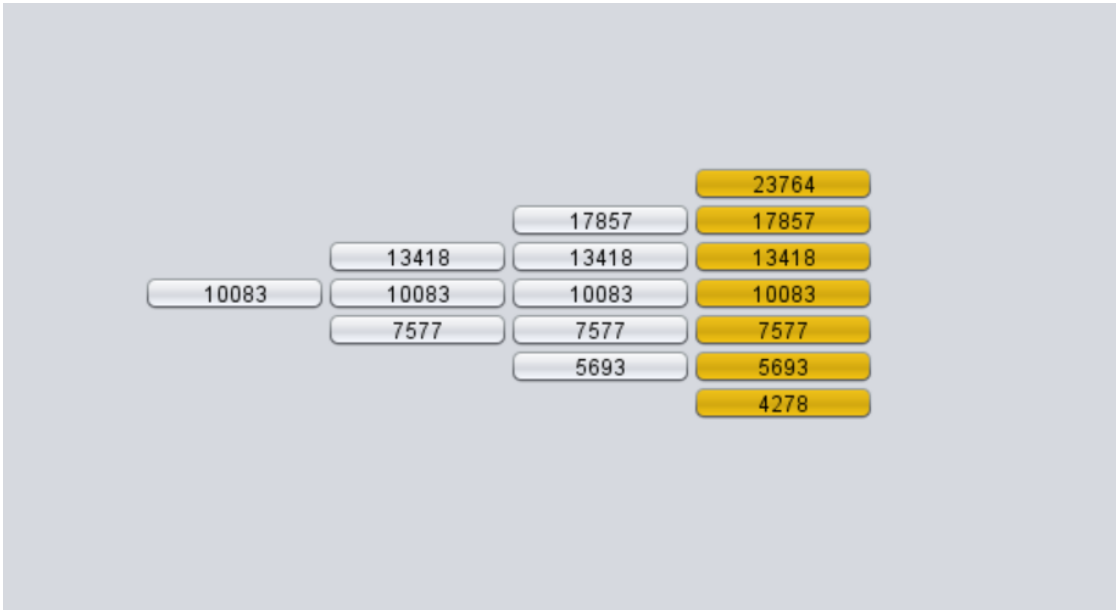
5.3.5 Skládání stromů

Důležitým požadavkem byla možnost skládání *binomických / trinomických* stromů za sebou. Hned na začátku máme možnost si vybrat, zda chceme vytvořit binomický či trinomický strom a taky, zda chceme vytvořit jednoduchý nebo skládaný strom, v případě skládaného je nutné mít hodnotu *stabilita*, která říká, z kolika období se má skládat jeden strom, tedy po kolika obdobích se v koncových nodech mají vytvořit úplně nové stromy, které budou potomkem toho prvního.

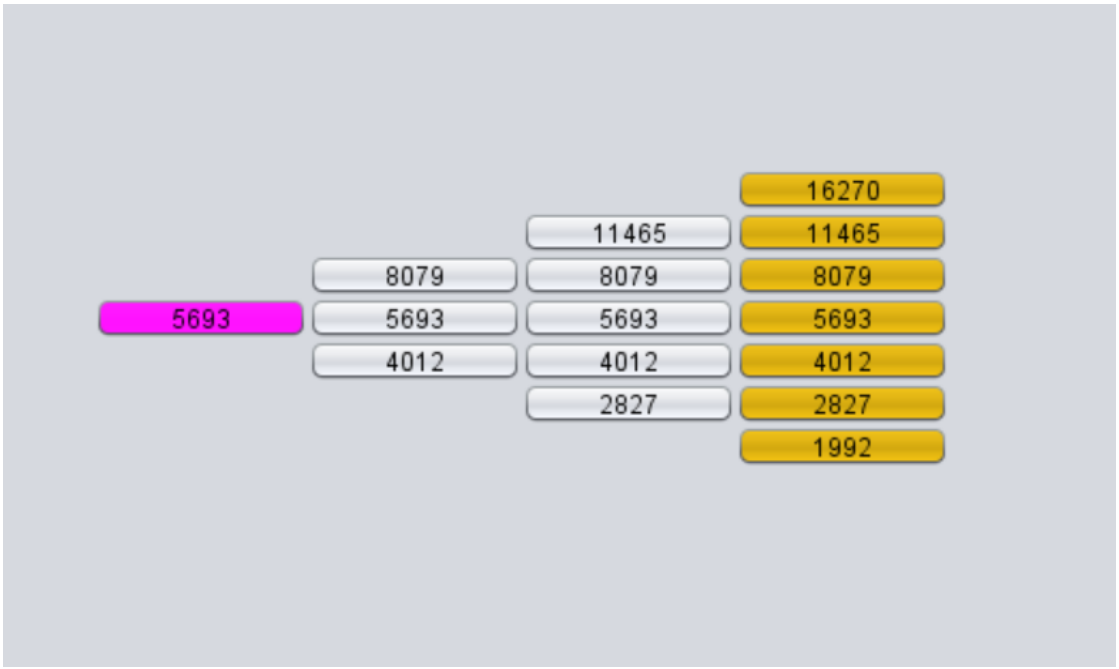
Jak to vypadá je vidět na obrázcích níže 5.4 a 5.5, kde jsou vidět dva po sobě jdou stromy. U stromu na obázku 5.5 je vidět, že jde o strom, který navazuje, protože první uzel je má fialovou barvu, a když se podíváme, tak vidíme, že hodnota 5693 je druhá zespoda u stromu na prvním obrázku.

Pro správnou interpretaci a výpočet hodnot bylo důležité si uvědomit, jaké hodnoty mají vstupovat do nově vytvářených stromů. Jde o stejné parametry jako na úplném začátku s tím rozdílem, že *hodnota podkladového aktiva* **S** bude odpovídat hodnotě v daném uzlu, ve kterém se vytváří nový podstrom. Podobná otázka se nabízí ohledně pravděpodobnosti, do jaké větve půjdeme. Můžeme buď stále přenášet tu původní nebo začít vždy od začátku.

Podstatné je, že podstromy se už dále navzájem nevidí, pro ně to vypadá, že nad nimi nic není. Pamatují si pouze odkaz na rodiče, kterému náleží, a rodič zase ví, na jaké úrovni jaký potomek je.



Obrázek 5.4: Skládaný strom 1/2



Obrázek 5.5: Skládaný strom 2/2

5.3.6 Uživatelské / Grafické rozhraní

Nedílnou součastí takovéto aplikace je grafické rozhraní, které uživatel uvidí a z kterého bude číst výsledky. Z toho důvodu by mělo být intuitivní, lehce ovladatelné a přehledné. Nejedná se o aplikace s jedním oknem, takže funkcionalita byla rozdělena do samostatných částí, které se dají otevírat z menu.

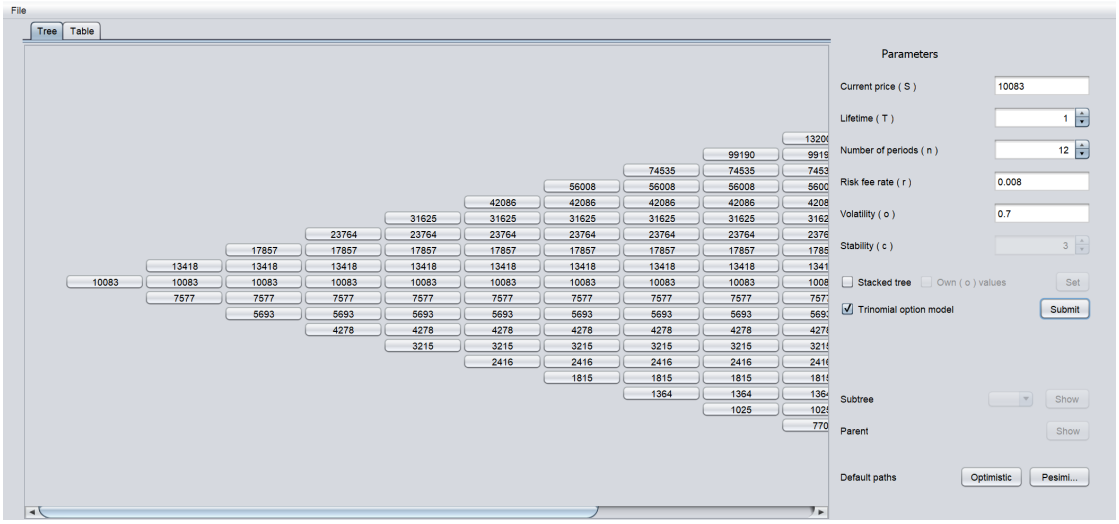
Dominantní částí grafického uživatelského rozhraní (**GUI**) jsou bezesporu velké vykreslovací plochy. Bylo potřeba najít vhodný způsop, jak zobrazovat vytvořené stromy tak, aby byly přehledné a aby se v nich uživatel mohl snadno orientovat. Pro tento účel byly použity nakonec dva způsoby, a to jeden grafický, kde je použit obyčejný *panel*, na který se v podobě vybraného stromu vygenerují a vykreslí jednotlivé uzly, a druhý způsob, kde není graficky vidět strom, ale je tam více podrobností a jednotlivé uzly jsou pomocí *DFS* algorotmu vypsány pod sebou v tabulce. V jednotlivých sloupcích jsou pak hodnoty daného atributu.

Nevýhodou, kterou bylo potřebova vyřešit, bylo vykreslování stromu, který se skládá z více podstromů, tedy že z posledních uzlů stromu vznikají nové stromy, které se překrývají, a není tedy možné je vykreslit najednou ve 2D prostoru. Z toho důvodu bylo rozhodnuto, že se vždycky vykreslí pouze jeden konkrétní strom. V případě, že v jeho koncových uzlech vznikají další stromy, tak tyto uzly jsou barevně zvýrazněny, podobně pouze jinou barvou je zvýrazněn i kořenový uzel stromu, pokud je daný vykreslený strom potomkem jiného, většího stromu.

Stejně důležitou částí je i vstupní formulář, přes který máme možnost zadat vstupní

hodnoty pro vytvoření nového stromu. Ten se nachází v pravé části okna vedle velké vykreslovací plochy.

Druhá velká vykreslovácí plocha se nachází v části, která pracuje s vybranými cestami. Funguje podobně jako plocha na hlavní obrazovce aplikace, s tím rozdílem, že nevykresluje už celý strom, ale pouze vybraný průchod. V právé části okna se nachází tentokrát nikoli formulář, ale seznam vytvořených cest, kde si můžeme vybrat, kterou cestu chceme zobrazit. Nad vykreslovací plochou se nachází ještě *taby*, které nám umožňují přepínání mezi podokny.



Obrázek 5.6: Uživatelské rozhraní

5.4 Technologie

Pro implementaci aplikace byla použita technologie Java a Java AWT

5.4.1 Java

Cílená platforma nebyla striktně zadaná a Java má tu velkou výhodu, že není platformně zavislá a dá se spustit kdekoliv. Další výhodou tohoto jazyka je jeho velké rozšíření a všeobecná známost, takže v případě, že by někdo k práci přišel, tak by s nějvětší pravděpodobností programovácímu jazyku bezproblému rozumněl. Z tohoto důvodu byla zvolena technologie Java pro vývoj tohoto nástroje.

5.4.2 Java AWT

Další otázkou bylo v čem udělat uživatelské grafické rozhraní (**GUI**). Jelikož u logiky byla vybrána Java a má se jednat o desktopovou aplikaci, bylo rozhodnuto že se nepůjde příliš daleko a zůstane se u nějakého Java frameworku pro grafiku, nakonec byl vybrát

základní grafický engine v Java, kterým je Java AWT. Ten poskytuje širokou paletu možností, z čeho lze vytvářet rozhraní, a také se jednotlivé komponenty dají relativně jednoduše přizpůsobit vlastním potřebovám.

5.5 Shrnutí

Tímto jsme si po teoretické stránce shrnuli, jak by měla požadovaná aplikace vypadat a hlavně jak by měla fungovat. Řekli jsme si, jak budeme řešit některé konkrétní problémy. Popsali jsme jednotlivé požadavky, které jsme si definovali a můžeme se posunout do další kapitoly, kde se podíváme na samotný postup implementace aplikace krok po kroku.

Kapitola 6

Implementace

Po seznámení se s jednotlivými částmi návrhu aplikace se můžeme přesunout k samotné implementaci aplikace. V první části kapitoly si popíšeme strukturu projektu a kde se co nachází. Pak přejdeme k nejrozsáhlejší části této kapitoly, a to k průběhu implementace aplikace, kde se postupně podíváme, jak byly implementovány jednotlivé části. Podíváme se na to, jaké problémy to obnášelo a s čím jsme se během práce potýkali. Na konci si ve stručnosti shrneme nasazení aplikace a to jak se celá implementace vlastně povedla a zda jsme zvládli všechno, co jsme chtěli.

6.1 Struktura projektu

V rychlosti si představíme strukturu projektu, jak je rozdělen do složek, a co které složky znamenají. Použity tu budou přesné názvy, které jsou použity v projektu.

1. **algorithsm** - V této složce se nachází kód k používaným algoritmům a třídám, které reprezentují nějaké akce

2. **listeners** - V této složce se dle názvu nachází *listenery*, tedy třídy, které poslouchají a reagují na určité akce, jako je stisknutí tlačítka, po kterém něco vykonají. Samostatná složka pro ně vznikla z důvodu zjednodušení čtení hlavních souborů, které díky těmto listenerům začínaly být nepřehledné, protože jde o relativně velké části kódu. Takhle jsou oddělené, což celkově působí přehledněji

3. **logic** - Jak název napovídá, tak se ve složce nachází nejdůležitejší logika aplikace. nachází se zde třídy, které definují chování a zacházení s objekty. Např. třída *TreeBuilder*, která je zodpovědná za vytvoření prázdného stromu. Nachází se tam i ostatní podobné třídy

4. **model** - Zde můžeme najít definice modelů jednotlivých použitých datových struktur od třídy, která definuje binomický či trinomický model, po třídy, které definují úplně základní strom

Zbylé soubory, které se nachází na stejné úrovni jako tyto složky, se dají označit za hlavní a jedná se o soubory, kde je implementována grafika a volá se v nich kód z ostatních složek.

6.2 Průběh implementace

Jako základní projekt byl použit stejný kód jako v mé bakalářské práci [9], a to kódy [21] a [22] z GitHubu, kde se nachází pod "GNU General Public License v3.0", a které dohromady tvoří jeden program, který počítá hodnotu evropské/americké opce. Kód byl od základu upraven pro potřeby této práce, a to nejdříve pro vytváření rekombinačního binomického modelu, posléze skládaného rekombinačního binomického modelu a v neposlední řadě také pro vytváření klasického a skládaného rekombinačního trinomického stromu.

Jelikož se navazovalo na předešlou práci [8], tak jsme se jí rozhodli nad rámec zádaní a požadavků začlenit do aplikace, kde se nachází jako jedna z položek v menu, která otevře okno s jednoduchým vstupním formulářem, který už známe z dané práce. Díky tomu lze v rámci jedné aplikace nalézt, jak to, co je zadáním této práce, tak se lze i jednoduše dostat k původní práci, kdyby si chtěl čtenář ověřit nebo porovnávat nějaké hodnoty pomocí metody reálných opcí či simulací.

6.2.1 Vytvoření stromu

To, jak samotné vytváření probíhá, bylo více popsáno v kapitole Analýza a návrh 5.3.3. Implementace této části funkcionality probíhala ze začátku bez nějakých zásadních problémů. Původní algoritmus se musel upravit na to, aby vytvářel rekombinační model, což u binomického modelu nebyl velký problém. V případě trinomického modelu se vyskytly menší problémy, protože původní algoritmus nepočítal s možností rekombinačního trinomického modelu a už vůbec ne s možností skládáného trinomického rekombinačního modelu. Největším oříškem byl fakt, že původní algoritmus byl navržen téměř s nejvyšší mírou abstrakce, které lze dosaáhnout, takže i ten normální nerekombinační model se skládá ze tří abstraktních rozhraní, která podle potřeby rozšiřují ta předchozí.

Samotné vytváření stromů probíhá rekurzivním volaním a použitím algoritmu *DFS*, který zajistí projití celého stromu. Největší změnou oproti původnímu stromu bylo, že jsme potřebovali model upravit tak, aby při průchodu z horního uzlu dolů nedošlo k vytvoření nového uzlu, ale k připojení již existujícího uzlu, který je navázán na spodní uzel, a tím nám mohl vzniknout rekombinační strom. To samotný algoritmus lehce komplikuje, ale ne natolik aby se nedal rozumně použít. Tímhle způsobem získáme prázdný strom složený z uzlů.

Další podstatnou změnou je, že když chceme v základu vytvořit strom složený z více stromů, tak voláme upravenou metodu na vytváření stromu, která bere parametr navíc, který určuje v jaké úrovni mají začínát nové stromy, a opět podle algoritmu *DFS* se postupně takto vytvoří celý složený strom. Musíme brát v potaz, že pokud zadáme nepřiměřeně vysoké hodnoty v životnosti projektu a počtu období, tak doba vytvoření

takového stromu může exponenciálně růst, obzlášť v případě složeného stromu, kde se bude vytvářet nepředstavitelně obrovské množství podstromů a kde pak narazíme na hranice výpočetního výkonu.

V dalším kroku musíme zavolat metodu *setTree*, která znova celý strom prochází algoritmem *DFS* a vypočítává vnitřní hodnoty jednotlivých atributů u každého uzlu.

6.2.2 Skládání stromů

Na základě požadavku na skládání binomických či trinomických modelů za sebe byla implementována funkcionalita skládaní takových stromů za sebou. Je to velmi užitečné v případě, že by nás zaujalo to, jaké máme teoreticky výhledy do budoucna, pokud bychom skončili za nějakou dobu v daném bodě. V tom případě by nám stačilo v tomto uzlu vytvořit nový strom, který začíná od tohoto bodu v čase a pokračuje dál. Implementovány jsou de facto dva možné přístupy, a to skládání nových stromů do koncových uzlů existujích stromů, nebo možnost definovat před vytvořením prvního stromu, že chceme vytvořit skládaný strom a zadat pouze, po kolika obdobích se mají ve všech koncových uzlech vytvořit nové podstromy. Druhý možný přístup je značně náročnější na výpočetní sílu pokud zadáme nějaký větší strom, protože narozdíl od první možnosti musí vytvořit všechny možné podstromy ve všech úrovních a všech podstromech. V první možnosti můžeme vytvořit jen ty podstromy, které nás zajímají, a tím i snížit náročnost úplného symetrického skládaného rekombinačního stromu o několika úrovních.

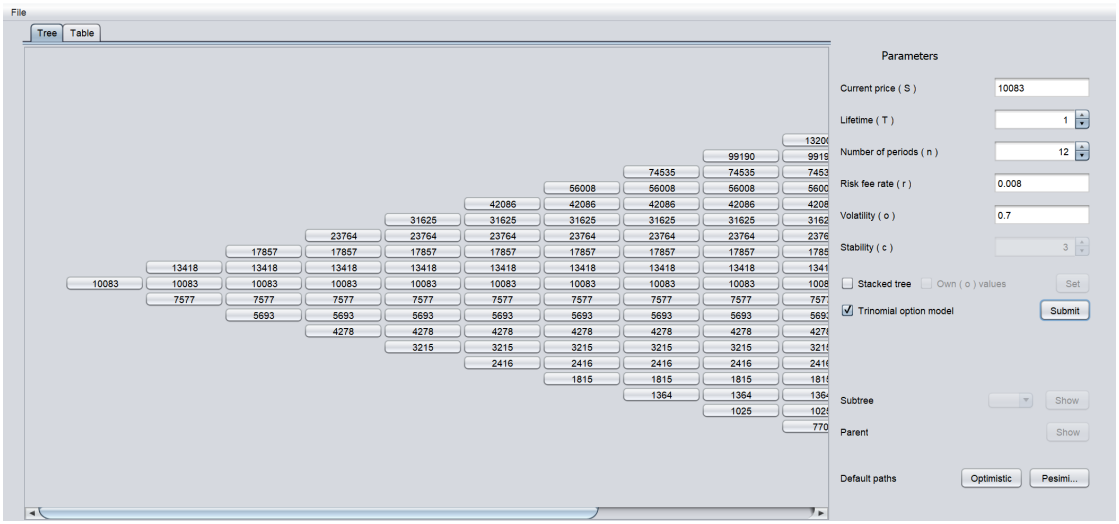
Zde se narazilo na problém, jak zajistit propojení mezi všemi vytvořenými stromy, tak aby se navzájem viděly a mohly si tak poskytovat navzájem informace. Řešením tohoto problému nakonec bylo vytvořením jakési abstraktní struktury nad fyzickou strukturou vytvoření stromu, kde jednotlivé stomy měly uzel *parent* a seznam uzlů *childrens*, pomocí kterého je už snadné přecházet mezi jednotlivými stromy. Největším oříškem v tuhle chvíli bylo udělat to tak, aby fyzická struktura měla přístup k datům uloženým v té abstraktní, přičemž samotný problém byl v tom, že abstrakce v původním řešení toto dosti znemožňovala.

6.2.3 GUI

Ve chvíli, kdy jsme byly schopni pomocí správně napsaných algorotmů a logiky vytvořit binomický či trinomický strom a vyplnit ho hodnotami, nezbývalo než se přesunout k dalšímu zásadnímu bodu, a to ke grafickému rozhraní (**GUI**), které je nezbytně nutné k tomu, aby se vytvořená data dala snadno interpretovat a zobrazit uživateli ve srozumitelné podobě. Jak vypadá hlavní panel grafického rozhraní, lze vidět na obrázku 6.1. Ostatní části grafického rozhraní včetně popisu i tohoto hlavního okna jsou rozebrány v dalších částech této kapitoly. Samotný návrh toho, jak by GUI mělo vypadat, vyšel ze zkušeností s návrhy uživatelských rozhraní tak, aby byl co nejjednodušeji použitelný pro nového uživatele a zároveň byl přehledný a dalo se v něm snadno a efektivně pracovat.

6.2.3.1 Hlavní okno

V první části se podíváme na hlavní okno 6.1, kde to celé začíná. Okno se skládá z několika seskládaných panelů *(JPanel)*, na kterých jsou umístěny další komponenty. V levé části je umístěný tabed panel *(JTabbedPane)*, který má dva taby, které tvoří scroll panely *(JScrollPane)*. Na prvním je umístěn další panel *(JPanel)*, který slouží jako vykreslovací plocha pro vytvořený strom. Na druhém se nachází tabulka *(JTable)*, která slouží k jinému výpisu stromu, a to v tabulkovém podání.



Obrázek 6.1: GUI - hlavní okno

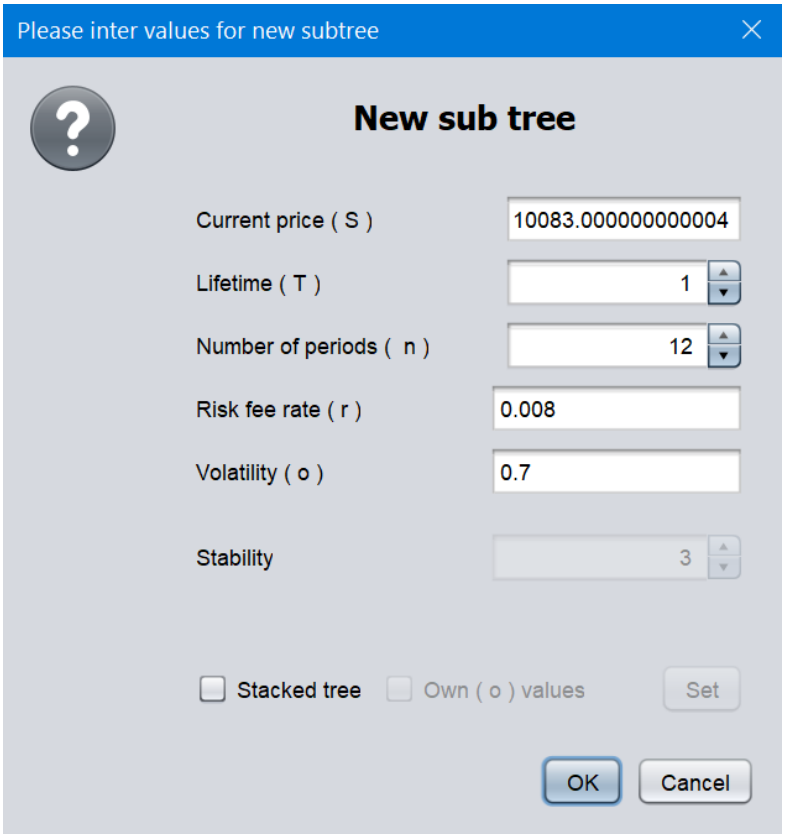
V pravé části je umístěn další panel *(JPanel)*, na kterém se nachází formulář, do kterého se zadávají hodnoty pro vytvoření prvotního stromu. Kromě políček pro zadání vstupních parametrů se zde nachází i nastavení toho, jaký strom chceme vytvořit, zda binomický nebo trinomický, normální nebo skládaný. U skládaného lze také ručně definovat, jak se bude vyvíjet volatilita v čase.

Na dalším obrázku 6.2 je vidět formulář, který uživatel uvidí při kliknutí na poslední uzel v daném stromě, pokud v tom uzlu neexistuje jiný strom. K tomu byl využit *JOptionPane*, který je volán *listenerem*, který se nachazí u daného uzlu.

Na posledním obrázku 6.3 můžeme vidět druhý možný způsob zobrazení vytvořeného stromu v podobě jednoduché a relativně přehledné tabulky, která umožnuje zobrazit více informací o daném uzlu. K tomu byla použita komponenta *JTable*, která nabízí přesně to, co potřebujeme. Pro posouvání mezi stromy v tomto zobrazení slouží tlačítka v dolní části pod formulářem pro vstupní parametry.

6.2.3.2 Okno pro práci s cestami

Druhým nejdůležitejším oknem je 6.4, které slouží k další práci s vybranými cestami a jejich mapováním na reálné konfigurace. Samotné okno vypadá na první pohled podobně



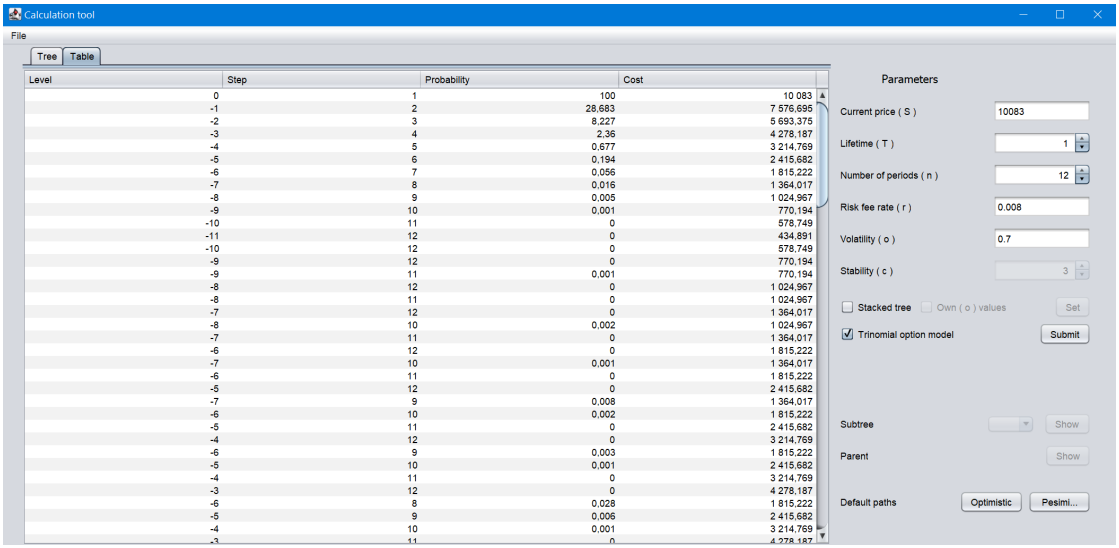
Obrázek 6.2: GUI - formulář na vytvoření nového podstromu

jako okno hlavní. Prvním rozdílem je to, že v pravé části se nenachází formulář pro parametry, ale místo něj se tam nachází seznam vytvořených cest. Pro tento seznam byla použita komponenta *JList*, která je umístěná v *JScrollPane*, a ten je položený na *JPanel*. To zaručuje, že se bude seznam automaticky rozšiřovat podle toho, jak velké okno bude, a bude se přizpůsobovat zvětšování okna aplikace. Takto je řešena většina částí aby, když uživatel roztáhne okno na větší rozměr, nezůstaly komponenty stejně malé a okolo se objevilo spoustu prázdného místa, takhle budou poměrově všechny komponenty zabírat stejné místo.

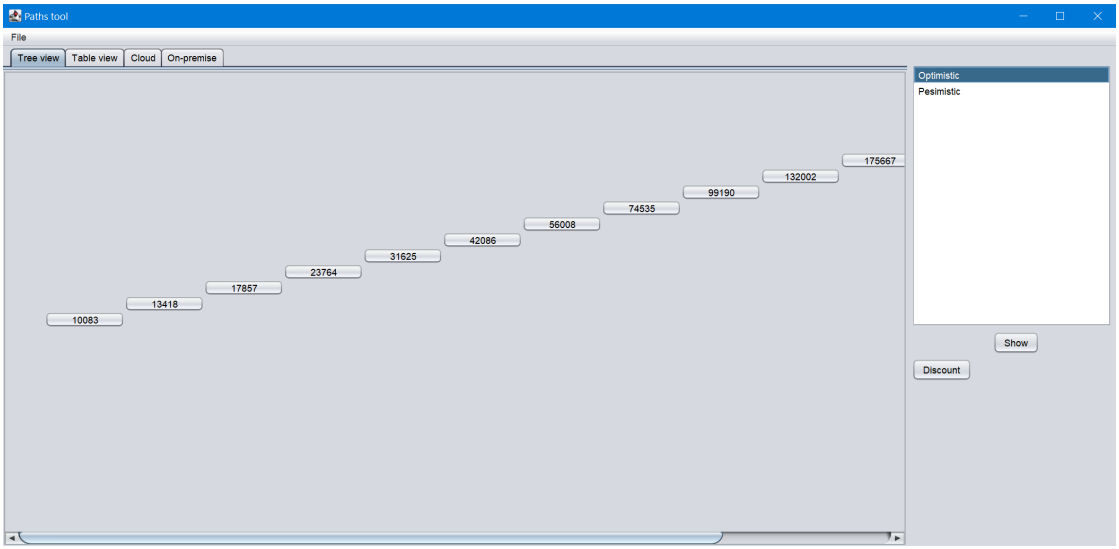
Dále se na tomto okně nachází další menu položka, která otevírá cestu do *Manažena konfigurací*, kterého si představíme v další části.

Nedílnou součástí je opět *JTabbedPane*, který nám poskytuje možnost mít více pohledů na vybranou cestu. Na prvním tabu se zobrazuje čistě vybraná cesta, na druhém tabu se zobrazuje daná cesta v podobě tabulky, jako je tomu i u hlavního okna. Poslední dva taby jsou zajímavější, protože tam už dochází k mapování vybrané cesty buď na *cloud* ve třetím tabu, či na *on-premise* ve čtvrtém tabu.

V tabu *cloud* či *on-premise*, když klikneme na jakoukoliv část cesty, tak nám to otevře



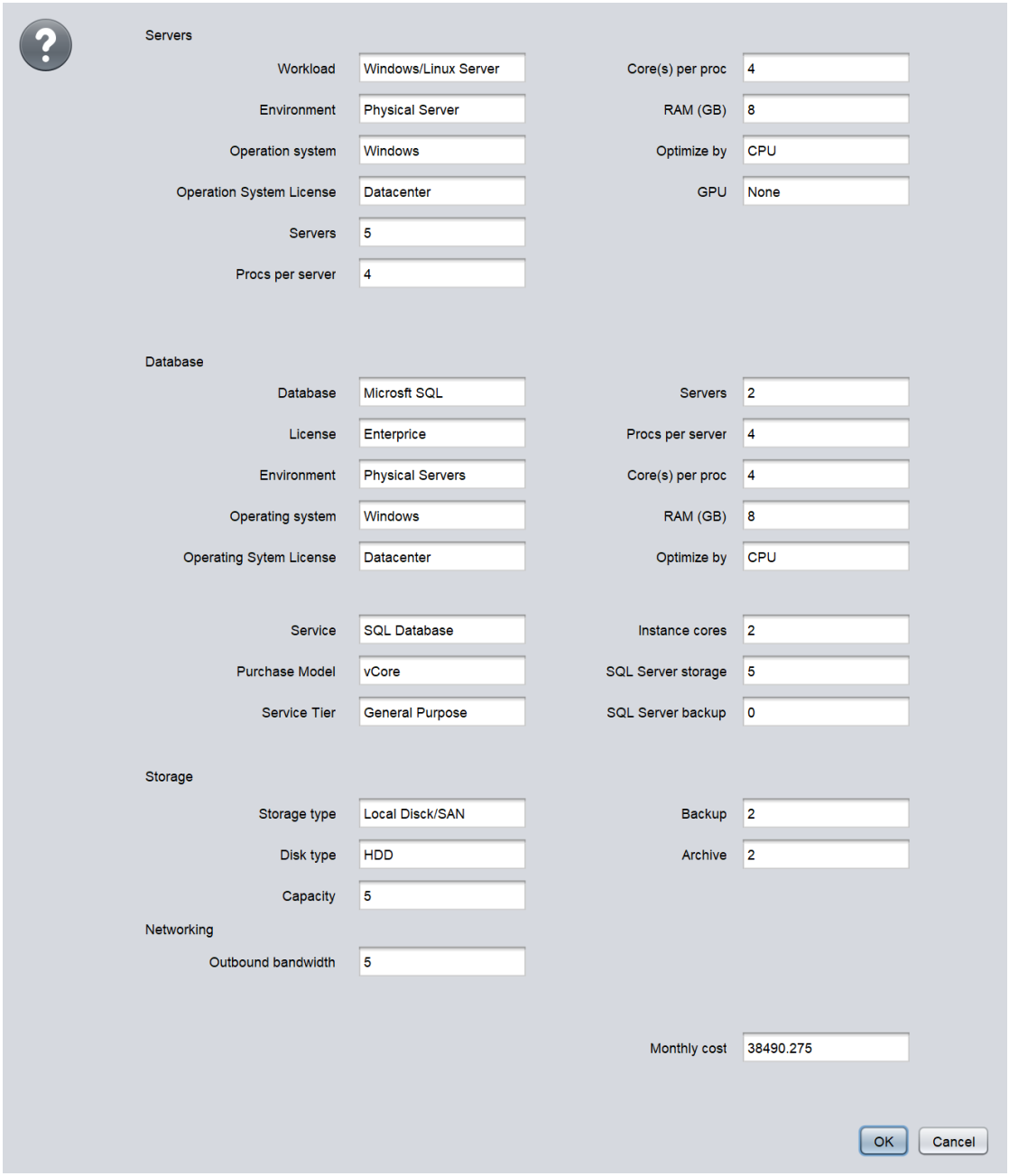
Obrázek 6.3: GUI - tabulkové zobrazení stromu



Obrázek 6.4: GUI - okno s průchody

okno 6.5, které nám ukáže nejlepší možnou konfiguraci, kterou si v danou chvíli můžeme dovolit, a úplně dole je vidět políčko *Monthly cost*, kde můžeme vidět, jaké jsou měsíční náklady na tuto konkrétní konfiguraci. Aby tato část fungovala co možná nejpřesněji, tak to vyžaduje kvalitní nástroj, který je schopný velmi dobře sestavovat tyto konfigurace, což je už nad rámec této konkrétní práce. Pro potřebu naší práce byl navrhnut relativně jednoduchý manažer konfigurací, kam lze ručně nebo ze souboru načítat konfigurace, z kterých pak při mapování program vybírá. V případě, že by byl náš nástroj napojen na

kalkulátor nákladů na IT infrastrukturu, jako má např. Microsoft Azure, dostali bychom přesnější výsledky, pro naší práci však bude stačit takto zjednodušený přístup.

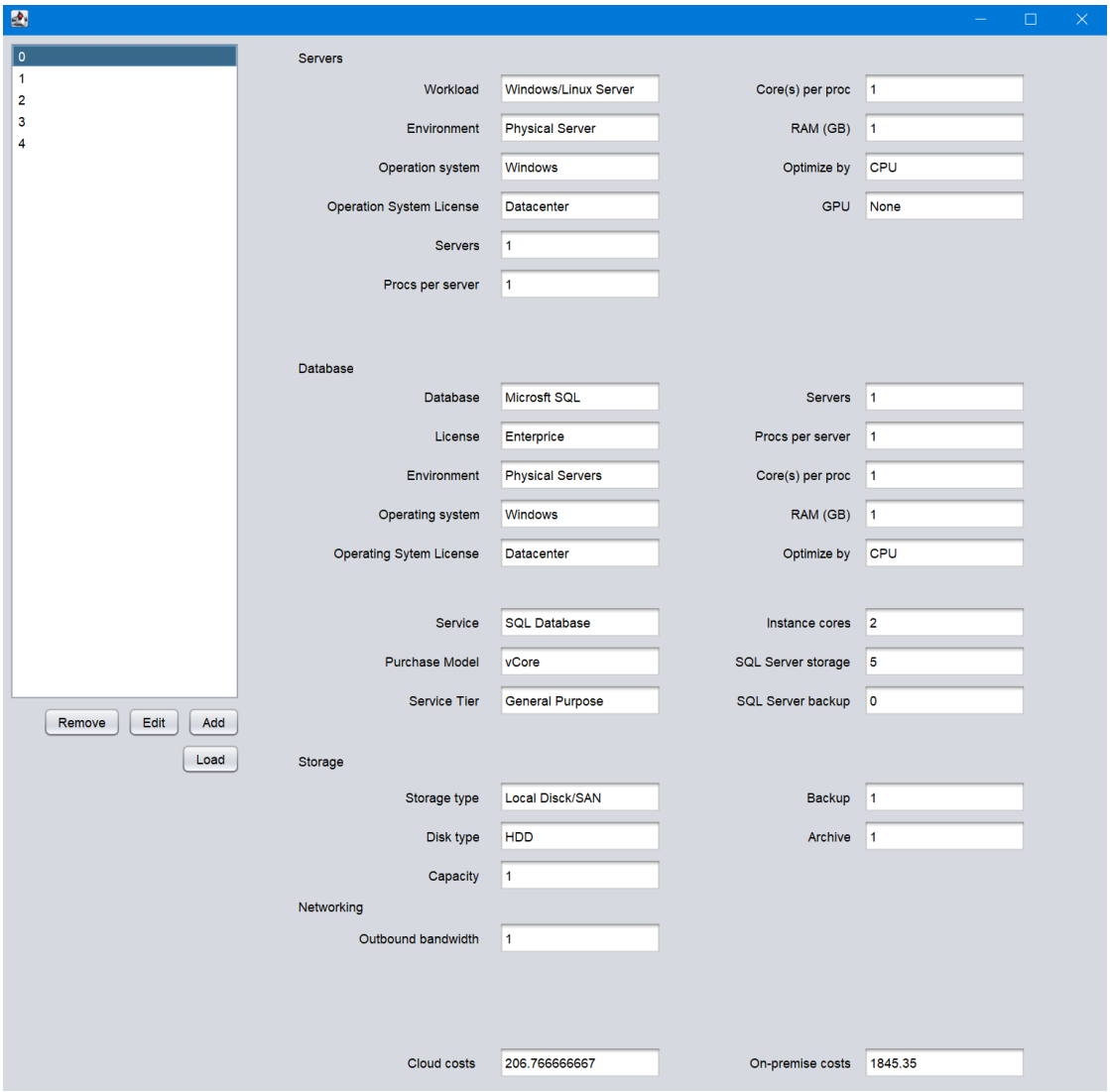


Obrázek 6.5: GUI - konfigurace

6.2.3.3 Manažer konfigurací

Posledním větším oknem je *Manžer konfigurací*, který lze vidět na obrázku 6.6. Zde je možnost přidávat, upravovat, mazat či rovnou načítat ze souboru konkrétní konfigurace, které se mají používat.

Celé okno se skládá z několika samostnatných částí. V levé části je vidět *JList*, který obsahuje seznam všech dostupných konfigurací. V pravé části se nachází *JPanel*, na kterém je postavený formulář, kde se zobrazuje aktuálně zvolená konfigurace.



Obrázek 6.6: GUI - manažer konfigurací

6.2.4 Save / Load

Jedním z funkčních požadavků byla možnost ukládání rozdělané práce nad stromem tak, aby bylo možné později přijít, načíst uloženou práci a pokračovat v ní dále. Pro tuto potřebu vznikl nový objekt *SaveLoadEncapsulation*, který si do sebe uloží ta nejdůležitejší data potřebná pro rekonstrukci stávající scény. Aby něco takového bylo vůbec možné, bylo třeba všechny objekty, které takto ukládáme, vytvořit jako *serializovatelné*, aby byly lehce převeditelné na bitový zápis, který se dá posléze jednoduše načíst zpátky a sestavit původní scénu, kde může uživatel dále pokračovat ve své práci.

6.2.5 Vytvoření průchodů stromem

Po vytvoření základu, tedy binomického či trinomického stromu, nastavení a dopočítání všech hodnot, přichází na řadu vytváření průchodů stromem. K tomu byl zvolen velice intuitivní a pro uživatele lehce pochopitelný způsob. Kliknutím na kořen hlavního stromu se změní stav aplikace na vytváření průchodů, což je indikováno změnou barvy tohoto uzlu. V tomto módu kliknutí na poslední uzel ve stromě nezavolá akci pro vytvoření nového stromu, místo toho se zavolá akce přidání cesty do seznamu již existujících průchodů.

Při výběru cesty usnadňuje přehlednost fakt, že byl implementován algoritmus, který po každém vybrání dalšího článku cesty všechny ostatní cesty, které se tím dostávají z dosahu výběru, znepřístupní pro přidání do cesty, protože by nedávalo smysl, kdyby to bylo možné. Když se nějaký uzel odebere, tak se zablokované cesty opět zpřístupní.

6.2.6 Konfigurace

Důležitou součastí jsou konfigurace, které se mapují na vybranou cestu stromem a na kterých lze vidět, jak se může vyvíjet naše IT infrastruktura v závislosti na tom, jak se firmě daří nebo nedaří. Viditelný by měl být rozdíl mezi *cloudem* a *on-premise* infrastukturou, který vychází z předchozích prací, a tím je vliv **flexibility**, protože v případě on-premise infrastruktury jednou zakoupený výkon nelze tak lehce snížit, protože nedává smysl prodávat servery, jelikož je nikdy neprodáme za stejnou cenu, za kterou jsme je koupili.

6.2.6.1 Mapování konfigurací

Velkou otázkou bylo, jak propojit hodnoty získané z binomického či trinomického stromu s konkrétní konfigurací IT infrastruktury v reálném životě. V případě *cloudu* to je jednodušší stejně jako v případě *metody reálných opcí*, protože stačí jenom najít konfiguraci, která se výší nákladů vejde do zisku společnosti v daném okamžiku. Naopak u *on-premise* infrastruktury se musí zohlednit fakt, že se vynakládají větší jednorázové investice do nového hardwaru a pak se platí provozní náklady. Jak rozložit tyto velké jednorázové investice, aby nezkreslovaly a daly se vůbec srovnávat s náklady u *cloudu* popisuje práce [12], kde je detailně vysvětleno, jak velké jednorázové investice rozložit pomocí anuit do měsíčních plateb, které jsou už porovnávatelné s poplatky za cloud.

6.2.6.2 Vytváření konfigurace

Kratší podkapitola se věnuje samotnému vytváření konfigurací. Aby bylo vůbec možné mapovat hodnoty z vytvořeného stromu na nějaké konfigurace, tak se ty konfigurace musí nějak vytvořit. V této části by mohla navazovat další práce, která by se věnovala tématu vytváření takových konfigurací a navrhla by algoritmus, který by fungoval např. podobně jako kalkulátor u Microfost Azure.

Při implementaci byla nutná volba, jak se k této problematice postavit. Vyvinout nějaký dokonalý a přesný nástroj, který by byl schopný takové konfigurace generovat, by bylo minimálně stejně náročné, jako tato práce, i proto byl zvolen zjednodušený přístup, který by měl být pro tuto práci dostačující, abychom zjistili, zda získáme výsledky, které budou dále použitelné.

Zvolený konfigurátor měl zahrnovat možnosti přidání, odebrání a úpravy konfigurací s tím, že pro jednoduchost by uměl i načítat větší množství konfigurací ze souboru. Místo algoritmu, který by generoval přesnější konfigurace, byla zvolena tato možnost, kdy konfigurace do aplikace ručně přidáme a vytvoříme tak určitou databázi konfigurací, s kterou pak aplikace pracuje.

6.2.7 Simulace

Stručně představíme nástroj zaměřující se na simulace a metodu reálných opcí. Jedná se o nástroj implementovaný v rámci bakalářské práce [9], a zde je přidán jen pro jednoduchost, když někdo bude aplikaci používat, tak aby byl po ruce i tento nástroj, ve kterém si lze například porovnávat hodnoty získané z hlavní aplikace.

6.3 Nasazení

Významnou výhodou je to, že aplikace napsaná v Java může být zabalená do souboru *.jar*, který je spustitelný v jakémkoliv prostředí a není nutná žádná instalace, protože funguje nezávisle na systému. Jediné, co stačí, je spustit přiložený jar a uživatel může pracovat. Výhodou tedy je, že ke spuštění není nutné používat konkrétní operační systém.

6.4 Zhodnocení

Implementace, místy přes větší či menší problémy, proběhla úspěšně. Byly implementovány všechny funkcionality definované v požadavcích. To, co se během implementace a vývoje ukázalo jako nepoužitelné či chybné, bylo po pravidelných konzultacích upraveno či uvedeno na pravou míru.

Kapitola 7

Testování

V této kapitole se zaměříme na testování nástroje, abychom zjistili, jaké výsledky dává a co výsledky vlastně říkají. Testovací data, jak bylo zmíněno i v ostatních pracech, není úplně snadné navrhnout, takže se budeme částečně i tady držet testovacích parametrů z jiných prací.

Kapitola je rozdělená na dvě větší části. První část se zabývá testováním binomického modelu a porovnává výsledky mezi různými průchody binomickým stromem, které simulují možné reálné scénáře vývoje. Nejlepší možná cesta, když všechno vychází dle plánů a daří se. Nejhorší možná cesta, když nevychází prakticky nic a prodělváme. A třetí průchod simuluje nejvíce reálnou cestu, kdy se chvíli daří, chvíli ne a nedosahuje žádných extrémů. K tomu je třeba ověřit vliv skládání jednotlibých stromů za sebou. Z toho důvodu se udělaly testy se stejnými vstupnímy parametry s tím rozdílem, že se zapla možnost, kdy to místo jednoho stromu vytvoří více stromů skládaných za sebou. Druhá část kapitoly se zabývá testováním trinomického modelu a jeho možností použití. Zajímá nás, jak trinomický model ovlivní výsledky, protože díky třetí možnosti zůstat na stejné úrovni nám poskytuje větší přesnost než model binomický. Detailněji je to rozdíl mezi binomickým a trinomickým modelem popsán v teoretické části této práce.

7.1 Vstupní data

V tabulce 7.1 můžeme vidět vstupní parametry, které budeme používát pro vytváření stromů. V řádcích kde je více možností, to znamená, že budeme testovat všechny kombinace, což vychází na minimálně 30 testů s tím, že variace volatility budou vždy zahrnuty v jedné tabulce.

Hodnoty parametrů, které se tu používají jsou vzaná z prací [7] a [20], kde k dané hodnoty stanovili jako použitelné pro testování. Samotné hodnoty parametrů se totiž neurčují jednoduše a vyžadují i expertní analýzu trhu například u parametru volatility, neboť to jak stabilní je trh je težké vyčíslit.

Tabulka 7.1: Vstupní data

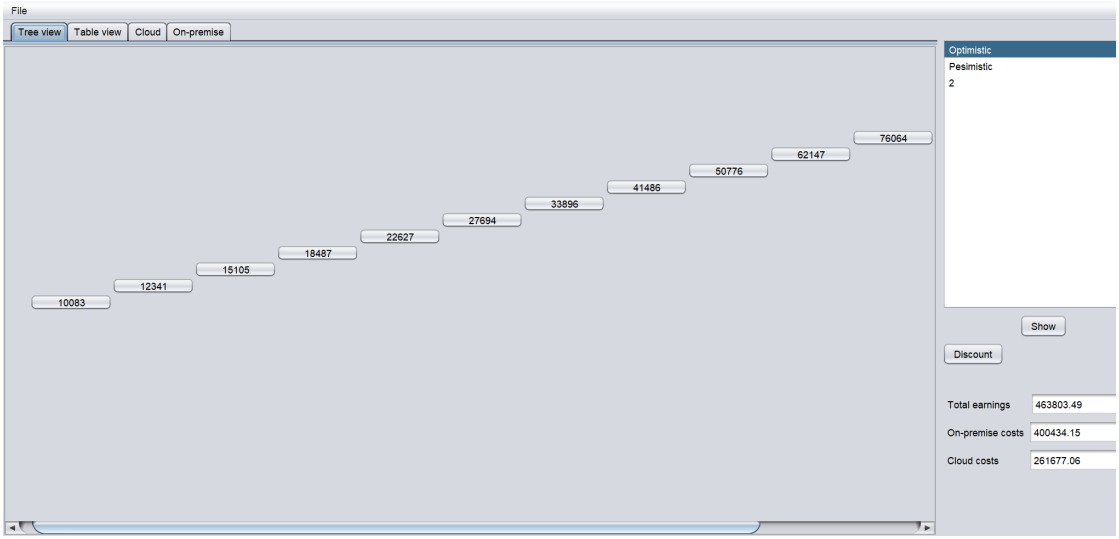
Vstupní data:	1	2	3	4	5	6
T	1					
n	12					
r	0,008					
σ^2	nízká	střední	vyšší		vysoká	
	0,1	0,3	0,5	0,7	0,9	1,1
Typ	jednoduchý			skládaný		
Cesta	nejlepší		nejhorší		průměrná	

1. **T** - životnost projektu v rocích
2. **n** - počet období, na které životnost rozdělíme
3. **r** - bezriziková úroková míra
4. σ^2 - volatilita
5. **Typ** - určuje, zda je pro výpočet použit jednoduchý strom nebo více stromů navázaných za sebe
6. **Cesta** - pro potřeby testování se určily tři typy cest, které se pro každé vstupní data porovnávají. Vybrány byly takto: nejlepší možná cesta, nejhorší možná cesta a jako poslední průměrná cesta, která více odpovídá realitě, protože jde nahoru a dolů a nedosahuje žádných extrémů

7.2 Binomický model

V první části se musel ověřit výpočet pro binomický model. Začalo se testy s jednoduchým stromem, kdy se vytvoří jeden strom s délkou odpovídající počtu období. Pro něj se zjistí a porovnají výsledky cest, které jsou popsány výše v tabulce vstupů 7.1. Po jednoduchém stromě se pokračovalo v testování se skládaným stromem, aby se ukázalo jaký dopad na výsledky toto nastavení bude mít.

7.2.1 Jednoduchý strom a optimistická cesta



Obrázek 7.1: Graf optimistické cesty v binomickém jednoduchém stromě

Vstupní data

Tento experiment se soustředí na to, jak je výpočet a následné mapování na konfigurace ovlivněno různou volatilitou, když se použije jednoduchý strom a vybere nejlepší možná cesta. Hodnoty jednotlivých parametrů lze vidět v tabulce 7.2. Provedeno bylo celkově šest testů. Testy byly prováděny pro 12 období v jednom roce. Kratší životnost projektu byla vybráno z důvodu, že potřebujeme také porovnat použití binomického modelu oproti modelu trinomickému a k tomu je potřebova simulovat stejné situace pro oba modely, a u trinomického modelu by výpočet u projektu, který má živostnost 2 roky a 24 obdoví trval dlouho kvůli větší komplexicitě trinomického modelu zvlášt u varianty, kdy se stromy skládají za sebe.

Vyhodnocení

Jak je vidět z tabulky, tak v případě kdy se volí nejlepší možná cesta, tak lze pravidelně navyšovat výpočetní výkon v obou případech, jak v cloudu, tak na on-premise, protože výnosy jsou dostatečně vysoké, aby pokryly náklady za obě možnosti. Celkové náklady na cloud vycházejí lépe než na on-premise. To způsobuje v tomto případě především korekce u on-premise, kde se započítávají do celkových nákladů i všechny splátky, které přesahují životnost projektu. Když by se dokoupil hardware měsíc před koncem projektu a bylo řečeno, že odepisujeme po 12 splátkách, tak se do celkových nákladů započte i zbývajích 11 splátek, které se nachází za koncem životnosti projektu. Kdyby tomu tak

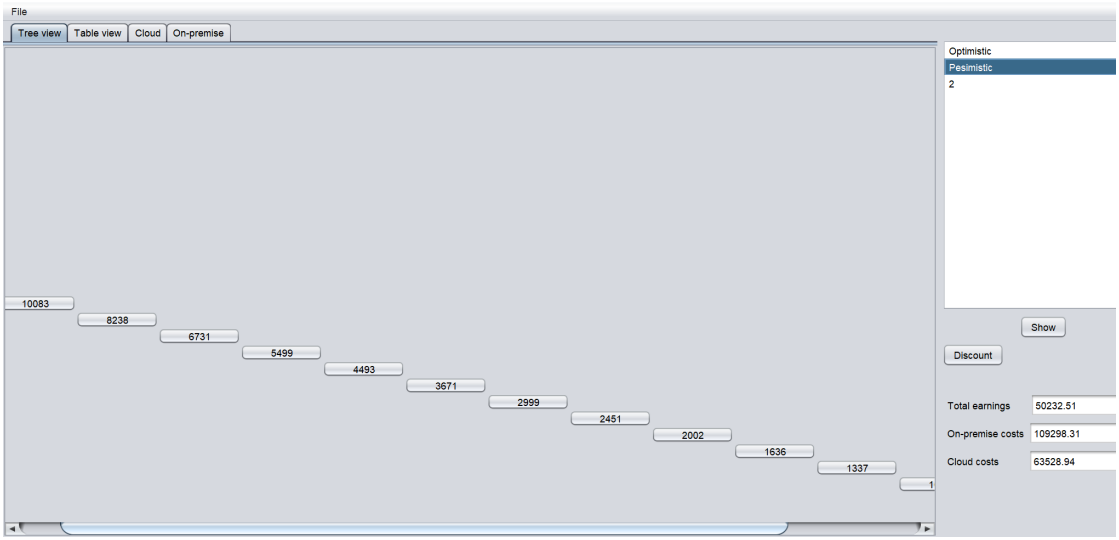
Tabulka 7.2: Experiment s jednoduchým binomickým stromem a nejlepší cestou

Test:	1	2	3	4	5	6
T	1					
n	12					
r	0,008					
σ^2	nízká	střední	vyšší		vysoká	
	0,1	0,3	0,5	0,7	0,9	1,1
Typ	**jednoduchý**					
Cesta	**nejlepší**					
Celkový zisk	142 520	203 641	302 101	463 803	733 900	1 191 717
Celkové on-premise náklady	181 586	154 262	423 108	400 434	377 759	377 759
Celkové náklady na cloud	118 672	118 672	175 874	261 677	209 277	318 878

nebylo, tak by byl on-premise v situacích, kdy se dokupuje výkon ke konci životnosti projektu dost zvýhodněn oproti cloudu a nebylo by možné je správně porovnat.

Vliv volatility lze vidět na zvyšujícím se celkovém zisku. To umožnuje více rozšiřovat infrastrukturu a tím se zvyšují i náklady na obě řešení.

7.2.2 Jednoduchý strom a pesimistická cesta



Obrázek 7.2: Graf pesimistické cesty v binomickém jednoduchém stromě

Vstupní data

Druhý experiment se zaměřuje podobně jako první na to, jak je výpočet a následné mapování na konfigurace ovlivněno různou volatilitou. V tomto případě se vybere jednoduchý strom a nejhorší cesta. Hodnoty jednotlivých parametrů lze vidět v tabulce 7.3. Provedeno bylo opět celkově šest testů pro 12 období v jednom roce.

Tabulka 7.3: Experiment s jednoduchým binomickým stromem a nejhorší cestou

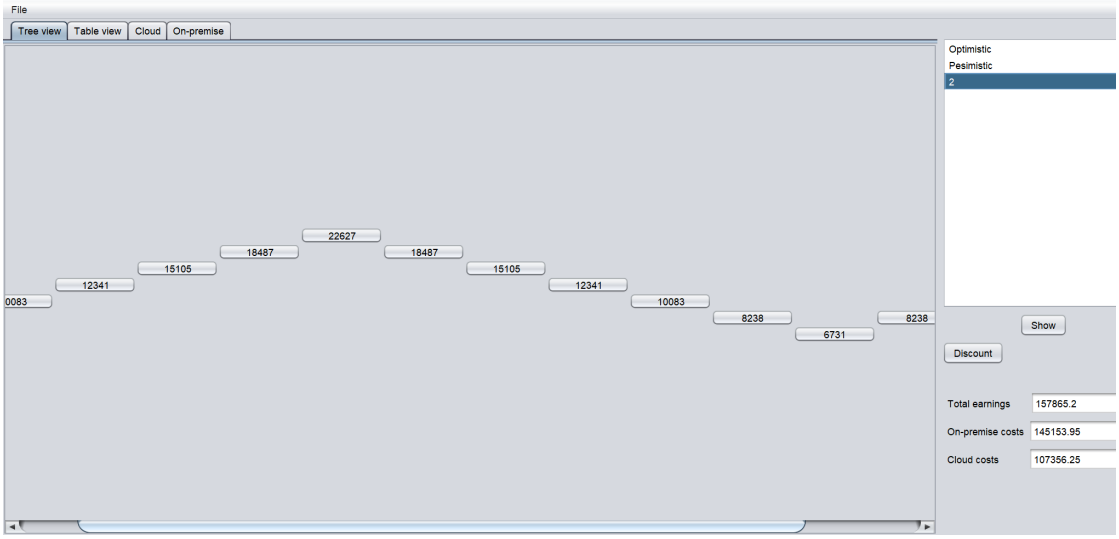
Test:	1	2	3	4	5	6
T	1					
n	12					
r	0,008					
σ^2	nízká	střední	vyšší		vysoká	
	0,1	0,3	0,5	**0,7**	0,9	1,1
Typ stromu	**jednoduchý**					
Cesta	**nejhorší**					
Celkový zisk	103 746	78 549	61 747	50 232	42 118	36 240
Celkové on-premise náklady	109 298	109 298	109 298	109 298	109 298	109 298
Celkové náklady na cloud	90 647	72 568	67 290	63 528	62 012	62 012

Vyhodnocení

Z tabulky 7.3 je vidět flexibilita cloudu oproti on-premise řešení. V tomto případě není možné povyšovat, a ideálním řešením je ponižování prvotní infrastruktury, která je každým obdobím více a více naddimenzovaná oproti požadavkům. U cloudového řešení s tím není problém a je vidět, že celkové náklady na cloud se snižují v závislosti na tom, jaké máme možnosti. Od určitého bodu však není dál možné ponižovat, protože infrastruktura aby byla schopná fungovat, tak nějaké minumum potřebuje. To je vidět na posledních dvou testech, kde už jsou celkové náklady za cloud stejné.

Různá volatilita zde způsobuje, že se buď celkové výnosy rychle zvyšují nebo naopak rychle klesají v závislosti na její velikosti. V případě nejhorší cesty jde o rychlé klesání, na kterém je vidět, jak se dané situaci dokáží přizpůsobit jednotlivá řešení, a které je v danou chvíli výhodnější.

7.2.3 Jednoduchý strom a průměrná cesta



Obrázek 7.3: Graf průměrné cesty v binomickém jednoduchém stromě

Vstupní data

Třetí experiment se zaměřuje podobně jako první dva na to, jak je výpočet a následné mapování na konfigurace ovlivněno různou volatilitou. V tomto případě se vybere jednoduchý strom a průměrná cesta, která nejlépe simuluje vývoj situace ve skutečném světě. Hodnoty jednotlivých parametrů lze vidět v tabulce 7.4. Provedeno bylo opět celkově šest testů pro 12 období v jednom roce.

Tabulka 7.4: Experiment s jednoduchým binomickým stromem a průměrnou cestou

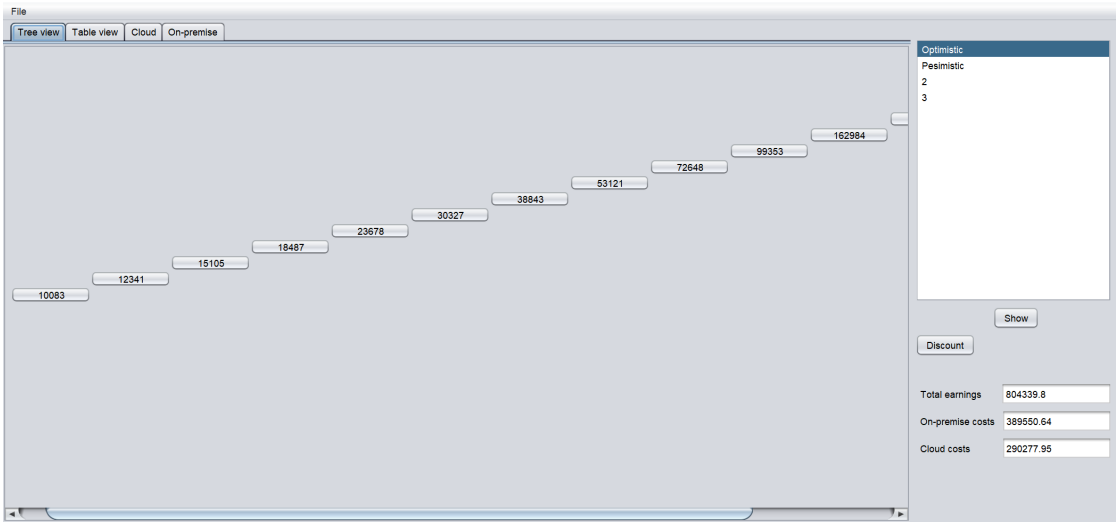
Test:	1	2	3	4	5	6
T	1					
n	12					
r	0,008					
σ^2	nízká	střední	vyšší		vysoká	
	0,1	0,3	0,5	**0,7**	0,9	1,1
Typ stromu	**jednoduchý**					
Cesta	**průměrná**					
Celkový zisk	124 704	133 514	144 443	157 865	174 241	194 139
Celkové on-premise náklady	109 298	154 262	145 153	145 153	377 759	377 759
Celkové náklady na cloud	114 090	114 090	111 845	107 356	105 839	105 839

Vyhodnocení

Výstupy testu jsou v tabulce 7.4. V tomto případě se ukázalo jak on-premise řešení zaostává za cloudem ve chvíli, kdy se situace na trhu dynamicky mění. V případě, kdy v jednom období jdou zisky nahoru a je potřeba navyšovat výkon infrastruktury a následně za pár měsíců jdou výnosy dolů a stačil menší výkon infrastruktury, je on-premise řešení velmi neohebné. To je vidět na celkových nákladech na on-premise, které uměrně s tím, jak se zvyšují výnosy a navyšuje výkon, rostou. Ve chvíli, kdy už tak velký výkon není potřeba, tak je on-premise infrastrutura nevyužitá a zvedá pouze náklady. Zatímco cloudové řešení, se těmto dynamickým změnám umí přizpůsobit, protože se platí jen za to, co se skutečně využije.

Z tabulky je také vidět, že celkové náklady na on-premise řešení přesahují celkový zisk, takže v této situaci je on-premise řešení nevýhodnou volbou.

7.2.4 Skládaný strom a optimistická cesta



Obrázek 7.4: Graf optimistické cesty v binomickém skládaném stromě

Vstupní data

Další sada tří experimentů experiment se zaměřuje jako první sada na to, jak je výpočet a následné mapování na konfigurace ovlivněno různou volatilitou. Znova byla simulovaná stejná situace, takže data zůstávají stejná. Zásadní rozdíl je, že místo jednoduchého stromu byl použit skládaný strom, aby bylo možné porovnat, jaký vliv má na výpočet použití jednoduchého či skládaného stromu. Vstupní hodnoty parametrů jsou vidět v tabulce 7.5.

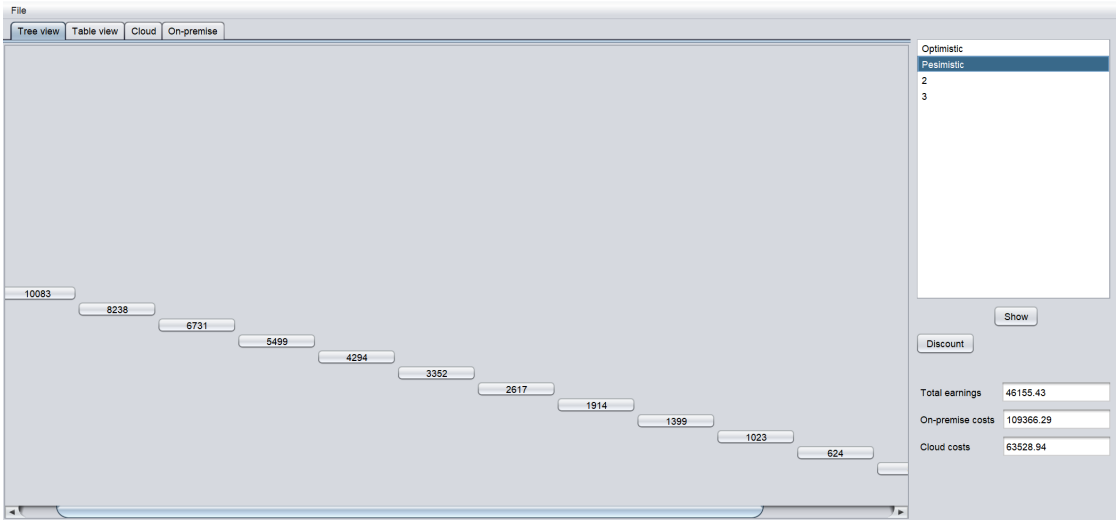
Tabulka 7.5: Experiment se skládaným binomickým stromem a nejlepší cestou

Test:	1	2	3	4	5	6
T	1					
n	12					
r	0,008					
σ^2	nízká	střední	vyšší		vysoká	
	0,1	0,3	0,5	**0,7**	0,9	1,1
Typ stromu	**skládaný**					
Cesta	**nejlepší**					
Celkový zisk	148 800	240 067	422 295	804 339	1 638 116	3 516 977
Celkové on-premise náklady	172 644	444 201	412 314	389 550	378 166	378 166
Celkové náklady na cloud	118 672	147 273	233 076	290 277	318 878	347 479

Vyhodnocení

Z tabulky 7.5 lze pozorovat, že vliv volatility je stejný jako u předchozích testů, což je předpokládané chování. V případě skládaného stromu pozorujeme vyšší hodnoty než u stromu jednoduchého. Na celkových nákladech na cloud a on-premise lze pozorovat stejné chování jako v případě s jednoduchým stromem. Pokles celkových nákladů na on-premise u vyšší volatility je způsoben korekcí, která se u on-premise provádí, a která už byla výše zmíněná.

7.2.5 Skládaný strom a pesimistická cesta



Obrázek 7.5: Graf pesimistické cesty v binomickém skládaném stromě

Vstupní data

Další test z druhé sady se zaměřuje jako předchozí testy na to, jak je výpočet a následné mapování na konfigurace ovlivněno různou volatilitou. Znova byla simulovaná stejná situace, takže data zůstávají stejná. V tomto případě se test zaměřuje na pesimistickou cestu. Vstupní hodnoty parametrů lze opět vidět v tabulce 7.6.

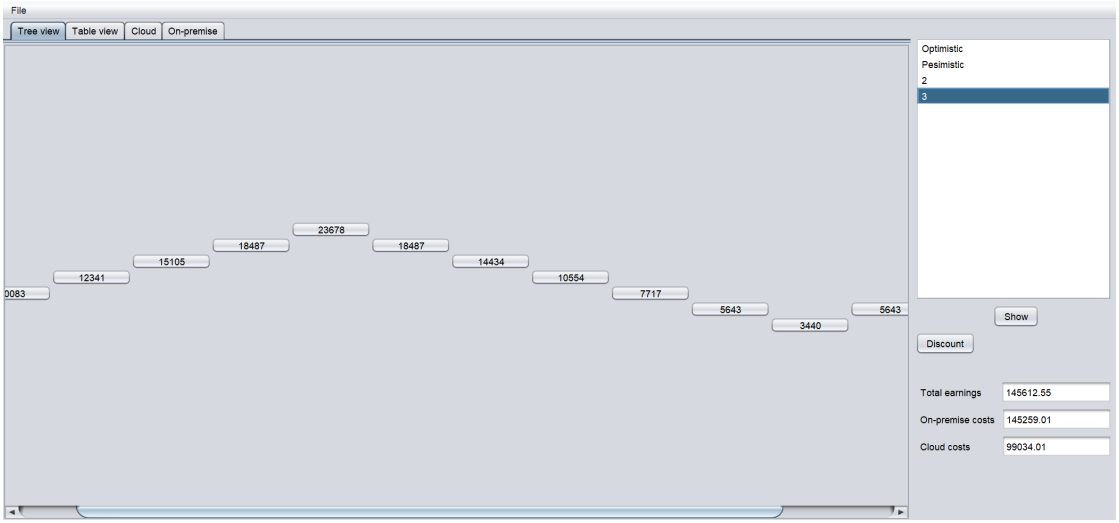
Tabulka 7.6: Experiment se skládaným binomickým stromem a nejhorší cestou

Test:	1	2	3	4	5	6
T	1					
n	12					
r	0,008					
σ^2	nízká	střední	vyšší		vysoká	
	0,1	0,3	0,5	**0,7**	0,9	1,1
Typ stromu	**skládaný**					
Cesta	**nejhorší**					
Celkový zisk	33 944	39 022	56 680	46 155	39 022	33 944
Celkové on-premise náklady	109 366	109 366	109 366	109 366	109 366	109 366
Celkové náklady na cloud	60 012	62 012	67 290	63 528	62 012	62 012

Vyhodnocení

Podle tabulky 7.6 je vidět, že vliv volatility je stejný jako u předchozích příkladů. On-premise náklady jsou vyšší než náklady na cloud. To je způsobeno nemožností ponížení u on-premise oproti cloudu, takže jde o předpokládané chování. Ukázalo se, že v případě skládaného stromu je jedno jaká bude volatilita, protože ve všech případech i náklady na cloud jsou vyšší než celkový zisk.

7.2.6 Skládaný strom a průměrná cesta



Obrázek 7.6: Graf průměrné cesty v binomickém skládaném stromě

Vstupní data

Posledním testem v druhé sadě a u binomického modelu se soutředí opět na to, jak je výpočet a následné mapování ovlivněno různou volatilitou. V tomto scénáři se vybrala průměrná cesta, tedy ta, která nejlépe simuluje chování skutečného světa. Vstupní hodnoty lze vidět v tabulce 7.7.

Tabulka 7.7: Experiment se skládaným binomickým stromem a průměrnou cestou

Test:	1	2	3	4	5	6
T	1					
n	12					
r	0,008					
σ^2	nízká	střední	vyšší		vysoká	
	0,1	0,3	0,5	**0,7**	0,9	1,1
Typ stromu	**skládaný**					
Cesta	**průměrná**					
Celkový zisk	122 200	126 871	134 610	145 259	160 222	178 946
Celkové on-premise náklady	172 644	154 378	145 259	145 259	378 166	378 166
Celkové náklady na cloud	112 562	105 828	102 067	99 034	99 034	132 185

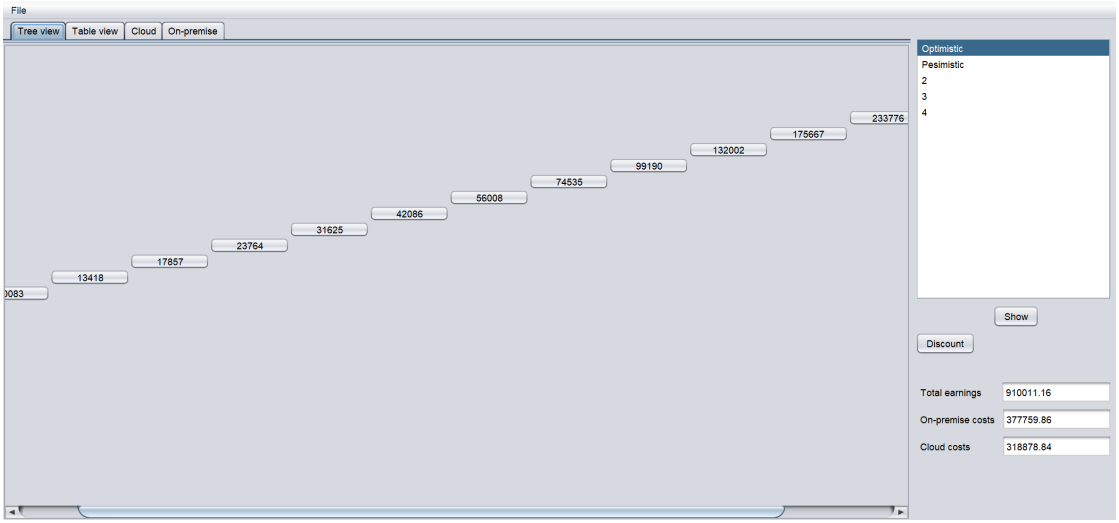
Vyhodnocení

Průměrná cesta i v případě skládaného stromu zachovává chování jako u jednoduchého stromu. Podle tabulky 7.7 je vidět, že tomu tak opravdu je. Čím vyšší je volatilita, tím jsou menší výnosy a vyšší ztráty, neboť u on-premise je nevyužitý výkon, za který platíme. Cloud se drží pod úrovní výnosů, takže v žádném okamžiku se neprodělává. To ukazuje, že výhoda flexibility cloudu tu skutečně je.

7.3 Trinomický model

V druhé části se musel ověřit výpočet pro trinomický model a tvrzení, že nám dá přesnější výsledky než model binomický. Aby bylo možné binomický a trinomický model porovnat, museli pracovat se stejnými vstupnímy daty, tedy pracovat se stejnými situacemi, aby bylo vidět, jaký rozdíl mezi nimi v daném okamžiku bude. Z toho důvodu byla ponechána vstupní data stejná a zaplo se akorát použití trinomického modelu místo modelu binomického.

7.3.1 Jednoduchý strom a optimistcká cesta



Obrázek 7.7: Graf optimistické cesty v trinomickém jednoduchém stromě

Vstupní data

V tabulce 7.8 jsou vidět vstupní data. Parametry byly zachovány stejné jako u testů binomického modelu, aby bylo možné oba modely porovnat. Postup testů bude taktéž podobný, takže první je sada s jednoduchým stromem a třemi vybranými typy cest. Druhá sada naopak se skládaným stromem.

Tabulka 7.8: Experiment s jednoduchým trinomickým stromem a nejlepší cestou

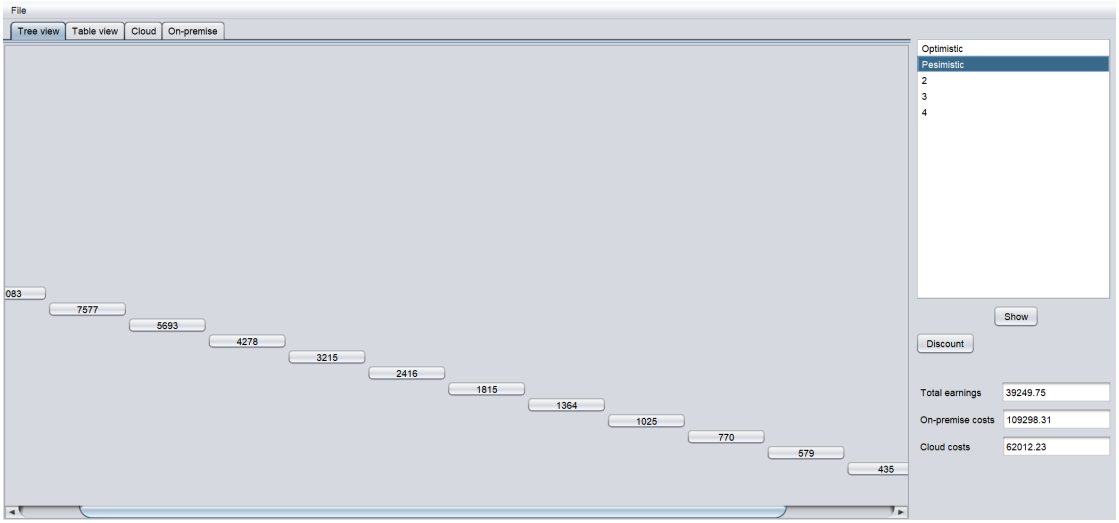
Test:	1	2	3	4	5	6
T	1					
n	12					
r	0,008					
σ^2	nízká	střední	vyšší		vysoká	
	0,1	0,3	0,5	**0,7**	0,9	1,1
Typ stromu	**jednoduchý**					
Cesta	**nejlepší**					
Celkový zisk	152 964	259 090	471 196	910 011	1 843 831	3 876 914
Celkové on-premise náklady	163 370	434 445	389 097	377 759	366 422	366 422
Celkové náklady na cloud	118 672	147 273	261 677	318 878	347 479	376 080

Vyhodnocení

Jak můžeme z tabulky 7.8 vidět, tak hodnoty jsou jiné než vychází v modelu binomickém. U nízké volatility vychází hodnoty velmi podobně, zatímco u vysoké volatility se dostáváme do násobně vyšších hodnot, to je způsobeno tím, jak funguje trinomický model, který byl představen v kapitole 3 v sekci 3.7. Kde je ukázáno, že jeden krok v trinomickém modelu se rovná třem krokům v modelu binomickém.

U nákladů můžeme vidět, že se nacházíme na stejným hodnotách jako u modelu binomického. To je způsobené malým množstvím konfigurací, které byly v době testování vloženy do aplikace, kdyby tam byly konfigurace zahrnující výkonější a nákladnější infrastruktutu, tak by samozřejmě rostly i náklady podle finančních možností mnohem více.

7.3.2 Jednoduchý strom a pesimistická cesta



Obrázek 7.8: Graf pesimistické cesty v trinomickém jednoduchém stromě

Vstupní data

Druhý test u trinomického modelu se zaměřil na normální strom a pesimistickou cestu. Data vychází z předchozím testů, a jak bylo zmíněno, zachovává se stejná modelá situace, aby na konci bylo možné porovnat binomický a trinomický model mezi sebou. Vstupní hodnoty jsou v tabulce 7.9.

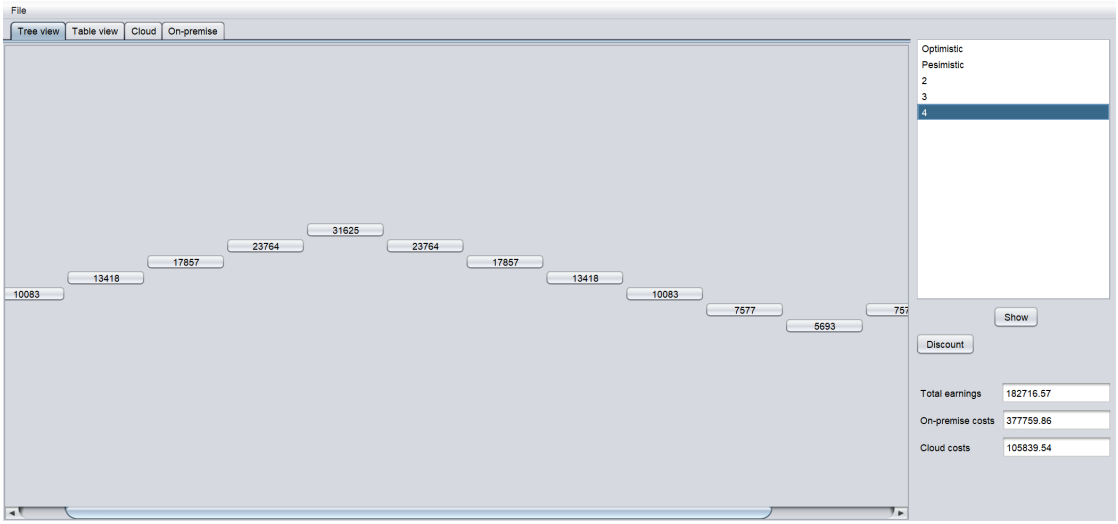
Tabulka 7.9: Experiment s jednoduchým trinomickým stromem a nejhorší cestou

Test:	1	2	3	4	5	6
T	1					
n	12					
r	0,008					
σ^2	nízká	střední	vyšší		vysoká	
	0,1	0,3	0,5	**0,7**	0,9	1,1
Typ stromu	**jednoduchý**					
Cesta	**nejhorší**					
Celkový zisk	97 624	67 353	49 894	39 249	32 393	27 743
Celkové on-premise náklady	109 298	109 298	109 298	109 298	109 298	109 298
Celkové náklady na cloud	86 157	68 807	63 528	62 012	62 012	77 179

Vyhodnocení

U nejhorší cesty i v případě trinomického modelu je vidět stejný vzorec chování jako u binomického modelu. Čím vyšší volatilita je, tím nižší jsou výnosy. Zároveň i náklady se chovají stejně jako předtím s tím rozdílem, že vychází přesnější výsledky. To potvrzuje teorii, že trinomický model umožnuje simulovat skutečný svět více přesně, než model binomický.

7.3.3 Jednoduchý strom a průměrná cesta



Obrázek 7.9: Graf průměrné cesty v trinomickém jednoduchém stromě

Vstupní data

Poslední test z první sady testů u trinomického modelu se zabývá průměrnou cestou za použití jednoduchého stromu a na to, jaký vliv na to bude mít různá volatilita. Vstupní data se nachází v tabulce 7.10.

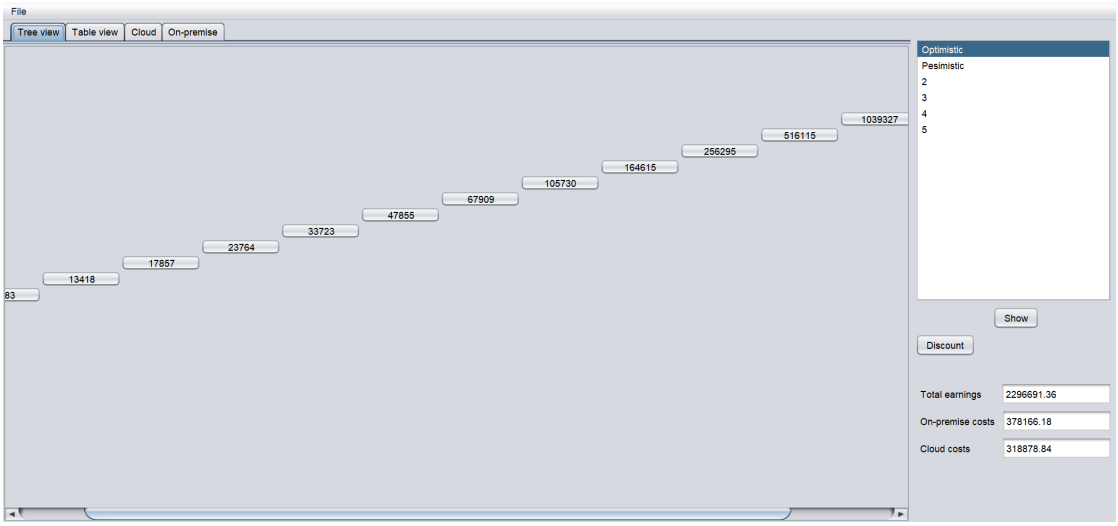
Tabulka 7.10: Experiment se jednoduchým trinomickým stromem a průměrnou cestou

Test:	1	2	3	4	5	6
T	1					
n	12					
r	0,008					
σ^2	nízká	střední	vyšší		vysoká	
	0,1	0,3	0,5	**0,7**	0,9	1,1
Typ stromu	**jednoduchý**					
Cesta	**průměrná**					
Celkový zisk	126 370	140 033	158 393	182 716	214 695	256 594
Celkové on-premise náklady	163 370	145 153	145 153	377 759	366 422	366 422
Celkové náklady na cloud	114 090	111 845	107 356	105 839	134 440	193 158

Vyhodnocení

Výstupy v tabulce 7.10 ukazují, že chování trinomického modelu se oproti binomického ani zde příliš neliší. I zde pozorujeme, že v případě on-premise řešení nastává situace, kdy se navýší výkon, v následujícím období se změní trh a sníží se potřeba tak velkého výkonu. To znamená, že on-premise infrastruktura je opět nevyužitá a dochází ke ztrátě. Zatímco cloud se dokáže přizpůsobit a snížit podle potřeby až na nutné minimum. To lze pozorovat na celkových nákladech jednotlivých řešení. U cloudu se ve všech případech drží pod úrovní celkového zisku, zatímco u on-premise je až na jednu výjimku vždy nad celkovým ziskem.

7.3.4 Skládaný strom a optimistická cesta



Obrázek 7.10: Graf optimistické cesty v trinomickém skládaném stromě

Vstupní data

Poslední sada testů se zaměřuje na skládané stromy v trinomickém modelu a nejlepší cestu. Podobně jako u předchozím testů se pozoruje vliv různé volatility na výpočet a mapování konfigurací. Hodnoty parametrů lze vidět v tabulce 7.11.

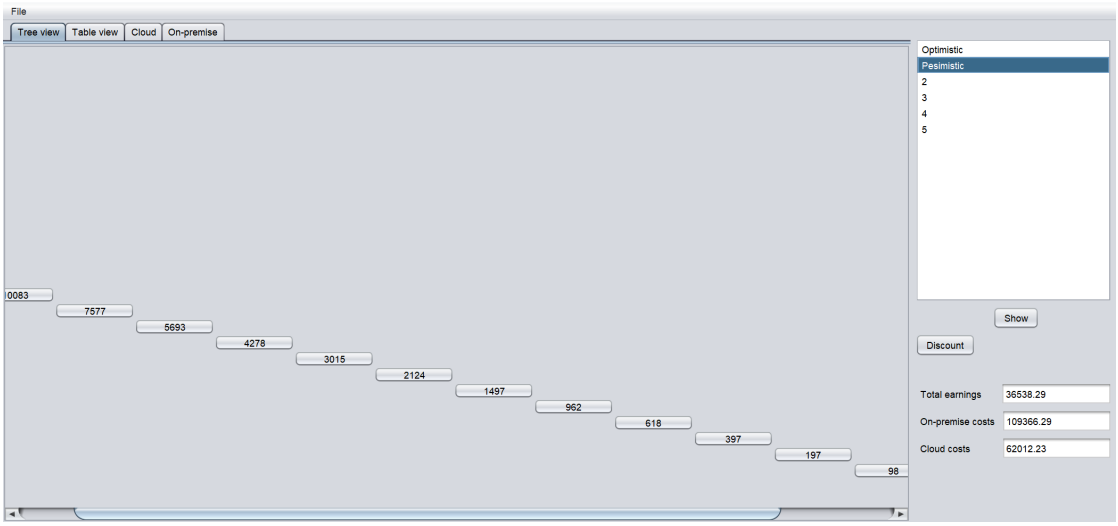
Tabulka 7.11: Experiment se skládaným trinomickým stromem a nejlepší cestou

Test:	1	2	3	4	5	6
T	1					
n	12					
r	0,008					
σ^2	nízká	střední	vyšší		vysoká	
	0,1	0,3	0,5	**0,7**	0,9	1,1
Typ stromu	**skládaný**					
Cesta	**nejlepší**					
Celkový zisk	163 123	337 625	824 051	2 296 691	7 030 624	2.29E7
Celkové on-premise náklady	163 516	423 693	389 550	378 166	366 781	366 781
Celkové náklady na cloud	118 672	204 475	290 277	318 878	347 479	376 080

Vyhodnocení

Výsledky lze vidět v tabulce 7.11. V tomto případě, kdy je použit trinomický model a skládaný strom, vychází nejvyšší hodnoty. Po prozkoumání je vidět, že chování nákladů odpovídá předpokladu. Cloud vychází nižší jelikož se jedná o optimistickou cestu a téměř v každém období lze navyšovat výkon. To vede k tomu, že korekce u on-premise do celkové ceny započítá všechny splátky za on-premise, které přesahují životnost. To je správný přístup, jinak by byl cloud znevýhodněn.

7.3.5 Skládaný strom a pesimistická cesta



Obrázek 7.11: Graf pesimistické cesty v trinomickém skládaném stromě

Vstupní data

Druhý test poslední sady se zaměřuje na skládané stromy v trinomickém modelu a nejhorší cestu. Podobně jako u předchozím testů se pozoruje vliv různé volatility na výpočet a mapování konfigurací. Hodnoty parametrů lze vidět v tabulce 7.12.

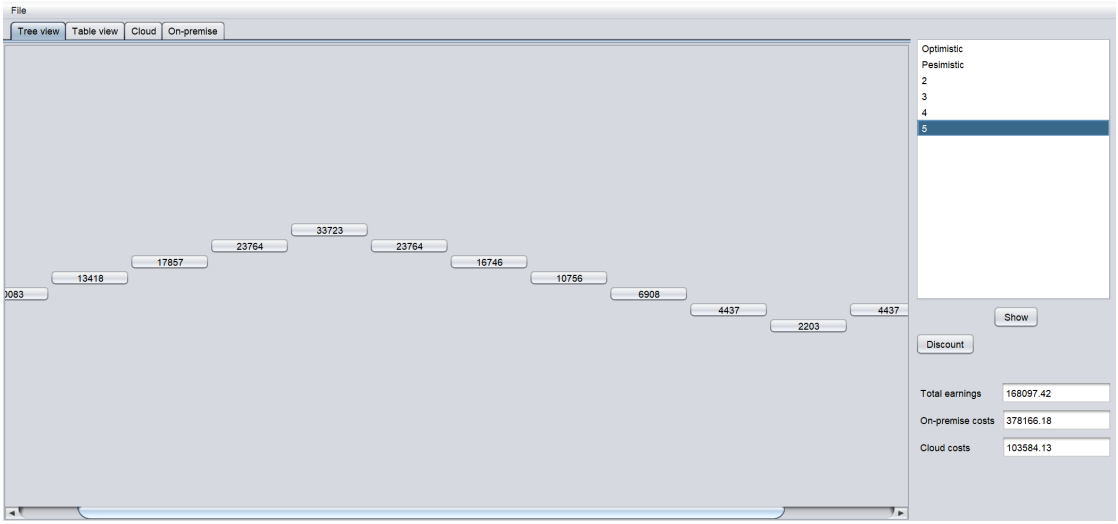
Tabulka 7.12: Experiment se skládaným trinomickým stromem a nejhorší cestou

Test:	1	2	3	4	5	6
T	1					
n	12					
r	0,008					
σ^2	nízká	střední	vyšší		vysoká	
	0,1	0,3	0,5	**0,7**	0,9	1,1
Typ stromu	**skládaný**					
Cesta	**nejhorší**					
Celkový zisk	93 482	62 012	45 854	36 538	30 631	26 601
Celkové on-premise náklady	109 366	109 366	109 366	109 366	109 366	109 366
Celkové náklady na cloud	83 124	67 290	63 528	60 012	62 012	77 179

Vyhodnocení

Z výsledků v tabulce 7.12 je nejlépe vidět, že vzorech chování se mezi oběma modely zachovává. Různá volatilita ovlivňuje výši výnosů a podle toho se musí přizpůsobit vybírané konfigurace v průběhu cesty. Cloud je znanatelně lepší než on-premise, přesto se však celkové náklady na cloud nachází nad celkovými zisky.

7.3.6 Skládaný strom a průměrná cesta



Obrázek 7.12: Graf průměrné cesty v trinomickém skládaném stromě

Vstupní data

Poslední test se zabývá trinomickým modelem se skládaným stromem a průměrnou cestou, který nejlépe simuluje skutečný svět. Vstupní data se nachází v tabulce 7.13.

Tabulka 7.13: Experiment se skládaným trinomických stromem a průměrnou cestou

Test:	1	2	3	4	5	6
T	1					
n	12					
r	0,008					
σ^2	nízká	střední	vyšší		vysoká	
	0,1	0,3	0,5	**0,7**	0,9	1,1
Typ stromu	**skládaný**					
Cesta	**průměrná**					
Celkový zisk	122 920	131 309	146 067	168 097	198 962	241 015
Celkové on-premise náklady	163 516	145 259	145 259	378 166	366 781	366 781
Celkové náklady na cloud	110 318	104 312	99 034	103 584	132 185	183 319

Vyhodnocení

Zde je opět vidět, že trinomický model zachovává chování binomického modelu a je přesnější. V tabulce 7.13 lze vidět výsledky, které jsou podobné těm, které vyšly u

binomického modelu. Taktéž lze pozorovat, jak se cloud přizpůsobuje aktuální situaci, zatímco on-premise postrádá flexibilitu, která by jí umožňovala rychle reagovat na změny trhu.

7.4 Vyhodnocení testování

V první řadě nás zajímalo, zda postup výpočtu uvedený v této práci zachovává trend, který můžeme pozorovat u reálných opcí a to, že flexibilita cloudu mu umožnuje se lépe přizpůsobovat dynamicky se měnící situaci na trhu. U cloudu lze jednoduše ponižovat a navyšovat výkon oproti on-premise, kde jednou nakoupený výkon težko snížíme, to lze vidět třeba u testu s binomickým modelem, jednoduchým stromem a průměrnou cestou v tabulce 7.2. Lze tedy říct, že uvedný typ výpočtu tento trend zachovává.

Potvrdil se nám, že je možné se podívat na náklady pro plánování informačních systémů i tímto způsobem, který je pro běžného uživatele čitelnější a pochopitelnější než komplikované nástroje pro výpočet reálných opcí.

Dále bylo možné si všimnout, že velký vliv na výpočet mělo nastavení typu stromu. V závislosti na tom, zda byl zvolen jednoduchý nebo skládaný strom, tak hodnota celkového zisku byla vyšší nebo nižší. Můžeme to pozorovat např. u testů z binomického modelu s nejlepší cestou a jednoduchým stromem 7.2 a skládaným stromem 7.5, kde lze vidět, že hodnota výnosů u vyšších volatilit se více liší, u nejvyšší volatility se bavíme o 1 191 717 jednotek vs 3 516 977 jednotek. Tento rozdíl mezi normálním a skládaným stromem je zapříčiněn vnitřními výpočty. U jednoduché stromu se jednou vypočtou všechny parametry nutné pro výpočet toho rozvoje, jak je vysvětleno v teoretické části práce, takže se vytváří jeden strom s délkou 12, když je životnost projektu rok. U skládaného stromu, když je nastavené, že každý strom má mít délku 3, tak se vytvoří 4 stromy s délkou 3 navázané na sebe s tím, že každý navazující strom si dopočítá parametry nutné pro rozvoj podle hodnot nacházejích se v bodě, kde končí předchozí strom.

Kapitola 8

Vyhodnocení stavu práce

8.1 Aktuální stav

Záměrem a cílem práce bylo pokračovat a prohloubit znalosti v problematice reálných opcí, flexibility cloud computingu a zároveň se na to podívat z trochu jiného úhlu pohledu. Usílí bylo směřováno k vytvoření nového podpůrného nástroje pro plánování nákladů na informační systém a k tomu částečně využít právě výše zmíněné metody a teorie. Celkově to nabízí alternativní přístup k pohledu na plánování nákladů na informační systém. V práci bylo zapotřebí navrhnout a implementovat daný nástroj, nutné bylo taky nástroj otestovat a zjistit, zda má vůbec smysl se tímto směrem dávat, nebo si říct, že tudy cesta nevede.

Tentokrát jsme šli dále a zaměřili se nejen na binomický model, ale také na model trinomický. Základem bylo navrhnout a vhodně upravit datové struktury a algoritmy pro každý typ modelu, neboť se od sebe liší a trinomický model je podstatně složitější než model binomický. I proto byly datové struktury pro přehlednost a lepší orientaci zcela odděleny a implementovány nezávisle na sobě.

Během cesty touto prací jsme se setkali s nejedním problémem, který jsme museli řešit. V první řadě jsme se museli zamyslet nad tím, zda dává smysl k tomu přistupovat způsobem, že z metody reálných opcí využijeme de facto jen binomický a trinomický model. Správnost tohoto přístupu však mohly ukázat jen testy, které nám buď mohly vrátit hodnoty, které dávají smysl a pak se jedná o správnou cestu, nebo hodnoty, které by nemusely dávat smysl, a pak by se nejednalo o správnou cestu.

Jedním z větších problémů se ukázaly být samotné konfigurace, neboť řešení které by bylo schopné navrhnout na základě nějaké částky ideální konfiguraci je samostatné téma. Proto bylo v práci využity zjednodušené řešení, kdy si konfigurace sami připravíme a nasypeme do nástroje, který s nimi pak dál pracuje a pamatuje si je.

Některé problémy zůstaly jako např. vstupní data, kde můžeme opět vycházet pouze z jiných prací nebo si nějaká vymyslet. Reálné příklady se k tomuto tématu stále obtížně hledají, jelikož se jedná stále o problematiku, která se zatím pohybuje více na akademické úrovni.

Také jsme se rozhodli k tomutu nástroji připojit i původní nástroj z bakalářské práce, protože se jedná o velmi podobnou problematiku a mohl by se hodit pro různé porovnávání, když bude někdo tento nástroj využívat.

V tuto chvíli je hlavním výstupem práce implementovaný nástroj, který může sloužit k dalším experimentům a úvahám jak využít tuto problematiku v praxi tak, aby mohla pomáhat podnikatelům a lidem zakládajícím si vlastní firmy.

8.2 Vyhodnocení hypotéz

Zde se vrátíme k hypotézám, které jsme si stanovili na úplném začátku práce a rozhodneme, které jsme potvrdili a které ne.

1. H_1 - **Trinomický model poskytuje reálnější (přesnější) informace než model binomický**

   Tato hypotéza se potvrdila, neboť to vychází už ze samotné definice binomického a trinomického modelu, vidět to může i na obrázku 3.6 v teoretická části o trinomickém modelu. To, že můžeme jít u trinomického stromu rovně, nám umožnuje se přiblížit situacím z reálného světa.

2. H_2 - **Rozdíl mezi *cloudem* a *on-premise* se chová stejně jako u *metody reálných opcí* či *simulací***

   Tato hypotéza byla potvrzena. Z testů 7.3.6, 7.3.3 a dalších je názorně vidět, že vzorec chování hodnot u metody reálných opcí, simulací i u našeho nástroje je stejný a zachovává rozdíl mezi cloudem a on-premise řešením.

3. H_3 - **Výsledná hodnota je srozumitelnější než v případě reálných opcí, kde nemusí každý vědět, co si pod danou hodnout má představit**

   I tato hypotéza byla potvrzena, s tím že uživatel vidí vybranou cestu 6.4 a konfigurace 6.5, tak si snáze představí, co dané hodnoty znamenají a lépe jim porozumí.

8.3 Možné rozšíření práce

Jak už bylo dříve naznačeno, tak autor během své práce narazil na pár místech na další témata, která by vydala na samostnou práci. Nejvýraznějším tématem, kde by bylo určitě možné navázat v jiné práci jsou konfigurace. Je totiž otázkou, jak nejlépe generovat konkrétní konfigurace IT infrastruktury na základě dané částky. Jednou z možností jsou nástroje podobné kalkulátoru nákladů od Microsoft Azure, který, kdyby se napojil na tuto aplikaci, mohl by pravděpodobně poskytovat ještě lepší a přesnější výsledky. Možností je samořejmě přijít i se zcela novým alternativním přístupem jak takové konfigurace vytvářet.

Kapitola 9

Metodika pro práci s aplikací

V této kapitole si představíme metodiku připravenou pro práci s aplikací, aby uživatel, který aplikaci nikdy neviděl, jí mohl začít bez problémů používat.

9.1 Příprava parametrů

V první řadě je potřeba znát hodnoty všech vstupních parametrů, která aplikace vyžaduje. Ty jsou nutné pro prvotní vytvoření binomického či trinomického modelu. Parametry, které do výpočtu vstupují jsou blíže vysvětleny v kapitolách 3.4, 3.6 a 3.7.

Některé parametry lze získat jednuduše. Najít si současnou cenu podkladové aktiva nebo životnost projektu není takový problém. Obtíž nastává u parametru *volatility*, který se už tak jednoduše nezískává. Blíže popsána je v kapitole 3.5 a 3.11.1, kde je popisován její vliv, a co to vlastně je. Další možnost je volatilitu odhadnout, k tomu nám může pomoci tabulka 9.1, v které lze vidět jaká hodnota volatility odpovídá jaké hodnotě. Na základě toho lze vybrat nejvhodnější hodnotu volatility.

	Hodnota volatility (σ)	Typický segment trhu
Žádná / nízká volatilita	0 - 0,20	Ustálený trh bez větších výkyvů, jako potraviny, banky, doprava.
Střední volatilita	0,20 - 0,30	Papírenství, elektronika, pohostinství, stavebnictví.
Vyšší volatilita	0,30 - 0,60	Počítače, reklama, výzkum a vývoj, lékařství, vzdělávání.
Vysoká volatilita	0,60 – 1,00	Většina internetových technologií a hi-tech produktů, e-commerce, těžba zlata, ropné vrty.

Obrázek 9.1: Rozdělení hodnot volatility [8]

Dále je třeba si projekt vhodně rozdělit na časové úseky. Samotný výpočet předpokládá období v délce jednoho měsíce, tedy počet období se bude rovnat násobku dvanácti v

závislosti na počtu let. Bezriziková úroková míra, se dá najít nebo výpočítatl.

Jako poslední zbývá se rozhodnout zda zvolit binomický či trinomický model a normální či skládaný strom. Trinomický oproti binomickému modelu dává přesnější výsledky. A rozdíl mezi normálním a skládaným stromem je popsán v kapitole 7.4.

9.2 Experimenty

Ve chvíli, kdy je vytvořený model, tak lze začít s experimenty, které ukážou, kdy se se, co vyplatí více. Doporučuje se vybrat nejlepší, nejhorší cesta a nějaké náhodné cesty, které mezi sebou lze porovnat, aby se ukázalo, kdy je lepší on-premise a kdy je lepší cloud. Jak takové cesty vypadají je vidět v experimentech 7.11 nebo 7.12, kde jsou vidět vstupní parametry a následně výsledky, které nástroj vrátí.

Experimentů je možné udělat kolik chceme a na všechny možné cesty, které lze vymyslet. Praktické bude vybírat cesty jako je nejlepší , nejhorší a ty pak doplnit několika obyčejnými průchody, které nedosahují extrémů.

9.3 Možnosti

V poslední části, kdy už jsou známé výsledky a lze říct, zda vychází lépe cloudové řešení nebo on-premise řešení, tak je ještě možné provést další ověření. Podle teorie 3 má největší vliv cena a volatilita, takže si lze ještě pohrát s těmito hodnotami. Vyzkoušet se dá ponížení volatility, tedy že bychom se přesunuli na stabilnější trh, a následně vidět, jak to ovlivní náš výsledek. V podobném duchu můžeme hýbat s hodnotou spotové ceny, aby bylo vidět, jaký vliv bude mít zisk.

9.4 Závěr

Cílem této kapitoly bylo přiblížit novému uživateli, který vidí nástroj poprvé, jak ho správně použít pro získání nejlepších výsledků. Před samotným výpočtem je potřeba znát hodnoty všech potřebným parametrů s tím, že některé se dají získat celkem snadno a jiné jako *volatilita* naopak celkem složitě. Po tomto úvodním nastavení lze vytvořit binomický nebo trinomický model, který je potřeba jako základ pro všechny další operace. Na modelu pak lze vytvořit tolik cest kolik bude nutné pro porovnání a vybrání vhodného řešení infrastruktury.

V poslední řadě lze ponižovat nebo povyšovat hodnotu volatility a spotové ceny, aby bylo vidět, zda vybrané řešení je vhodné i při změně trhu nebo v závislosti na nákladech.

Závěr

Smyslem této práce bylo navázat na již existující práce týkající se reálných opcí a cloud computingu. Zadáním práce bylo vyvinout nový nástroj pro podporu plánování nákladů na informační systém. Cílem bylo zakomponovat do tohoto nástroje kromě binomického modelu i model trinomický, který by měl poskytnout přesnější výsledky. Celkově se práce skládá z více menších samostných celků - nástroj na vytváření a vykreslování stromů, nástroj pro práci s vytvořenými cestami a nástroj pro vytváření a úpravy konfigurací. K tomu byly vytvořeny také menší funkcionality, jako je ukládání rozdělané práce.

V první řadě byly představeny nutné teoretické základy pro pochopení reálných opcí, cloud computingu a spojení mezi nimi. Představeny byly vzorce, na kterých celá teorie stojí, včetně vzorců pro vytváření binomického či trinomického stromu. Na to byla představena úvaha o mapování hodnot vycházejích z binomického nebo trinomického modelu na reálné konfigurace. Z toho všeho byly specifikovány požadavky na výsledkou aplikaci, která byla následně implementována a otestována. V praktické části práce je možné se podívat na samostný postup implementace a na to, jak se postupovalo krok po kroku až k výsledné podobě nástroje.

Nejzásadnějším výstupem této práce je implementovaný nástroj, který by díky jeho jednoduchosti a přehlednosti měl zvládnout použít každý, koho daná problematika zajímá. I tady se jedná zatím o nástroj použitelný spíše v akademické sféře pro různé experimenty. Lze však předpokládat, že jednoho dne bude takový nástroj samozřejmostí pro každého, kdo vede větší firmu. Na tuto práci se dá v mnoha ohledech dále navazovat či zdokonalovat už připravené postupy.

Z vlastního pohledu byla práce velkým přínosem, protože se stále jedná o relativně málo zdokumentovanou problematiku a je třeba zapojit vlastní rozum, být tak trochu průkopníkem a podívat se do míst, kde jiní ještě třeba nebyli, a zjistit, zda je to cesta, kterou má smysl se vydat. Z technologického hledisla byla pro mě přínosem práce s Java AWT, což by se mohlo v budoucnu hodit, jelikož to má relativně široké pole uplatnění, neboť si člověk může většinu komponent celkem jednoduše přizpůsobit podle svých představ a sestavit z nich, co chce. Celkově tedy práci hodnotím pozitivně.

Literatura

[1] Starý, O.: *Reálné opce*. Praha: A Plus, 2003.

[2] KODUKULA, P. a. C. P.: *Project Valuation Using Real Options: A Practitioner's Guide*. Pine Island Rd.: J. Ross Publishing, 2006.

[3] NÁPLAVA, P.: *Evaluation of Cloud Computing Hidden Benefits by Using Real Options Analysis*. Acta Informatica Pragensia, 2016.

[4] Abrahamová, M.: *Binomický a trinomický model oceňování opcí*. Západočeská univerzita v Plzni, 2015.

[5] NÁPLAVA, P.: *Cloud computing a reálné opce jako akcelerátor začínajících IT technologických firem*. Acta Informatica Pragensia, 2014.

[6] Microsft: Microsoft Azure. `https://azure.microsoft.com/cs-cz/pricing/tco/calculator/`, (accessed: 12.04.2021).

[7] Trnka, V.: *Nástroj pro podporu výpočtu hodnoty reálných opcí*. České vysoké učení technické v Praze, 2017.

[8] Scholleová, H.: *Hodnota flexibility. Reálné opce*. C. H. Beck, první vydání, 2007.

[9] Brzobohatý, M.: *HODNOCENÍ FLEXIBILITY PROVOZU APLIKACÍ V PROSTŘEDÍ CLOUD COMPUTINGU*. ČVUT Praha, 2019.

[10] Plummer, D. C.; Bittman, T. J.; Austin, T.; aj.: *Cloud Computing: Defining and Describing an Emerging Phenomen*. Gartner Analysis, 2008.

[11] Plummer, D. C.; Smith, D. M.; Bittman, T. J.; aj.: *Five Refining Attributes of Public and Private Cloud Computing*. Gartner Research, 2009.

[12] Náplava, P.: *Reálné opce a jejich využití v prostředí IT technologií*. České vysoké učení technické v Praze, 2017.

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

[14] Ambrož, L.: *Oceňování opcí*. C. H. Beck, první vydání, 2002.

[15] COX, S. A. R. a. M. R., John C.: *Option pricing: A simplified approach*. Journal of Financial Economics, 1979.

[16] HAHROKHSHAHI, Z.: *HOW MAKE REAL OPTION DEPENDABLE AND UNDERSTANDABLE AS A STRATEGIC DECISION MAKING TOOL?* ACRN Oxford Journal of Finance and Risk Perspectives, 2016.

[17] MYERS, S. C.: *Determinants of corporate borrowing*. Journal of Financial Economics, 1977.

[18] PATERNOSTER, C. G. a. M. U. a. T. G. a. P. A., Nicolò: *Software Development in Startup Companies: A Systematic Mapping Study*. Inf. Softw. Technol, 2014.

[19] YAM, A. B. a. S. S. a. C. I., Chew Yean: *Migration to cloud as real option: Investment decision under uncertainty*. Proc. 10th IEEE Int. Conf. on Trust, Security and Privacy in Computing and Literatura 138 Communications, TrustCom 2011, 8th IEEE Int. Conf. on Embedded Software and Systems, ICESS 2011, 6th Int. Conf. on FCST, 2011.

[20] Mastný, M.: *Real Option Analysis and cloud computing investments*. Czech Technical University in Prague, Faculty of Electrical Engineering, 2017.

[21] Dinter, D.: java-coxrossrubinstein. `https://github.com/danieldinter/java-coxrossrubinstein`, Apr 2016, (accessed: 12.04.2021).

[22] Dinter, D.: java-treemodel-dfs. `https://github.com/danieldinter/java-treemodel-dfs`, Apr 2016, (accessed: 12.04.2021).

Příloha **A**

Seznam použitých zkratek

GUI Graphical user interface

On-premise model infrastruktury, ve kterém si veškerý hardware pořídíme přímo k sobě, takže máme nějakou serverovnu

Nejistota stav, kdy neznáme všechny možné důsledky svého rozhodnutí a pravděpodobnosti toho, že nastanou

NPV hodnota, reprezentující celkovou současnou hodnotu všech peněžních toků

Náklad představuje spotřebování ekonomického zdroje, které je obvykle spojeno se souběžným nebo budoucím výdejem peněz

Metoda reálných opcí je metoda pro hodnocení investic, která bere v úvahu riziko i flexibilitu

Flexibilita schopnost pružně reagovat na měnící se podmínky, přizpůsobivost

Binomický model nespojitý model oceňování opcí, který předpokládá, že se cena podkladového aktiva mění diskrétním způsobem

Trinomický model nespojitý model oceňování opcí, který předpokládá, že se cena podkladového aktiva mění diskrétním způsobem a má o jednu možnou cestu navíc oproti binomickému modelu

Anuita konstantní platba po smluvenou dobu. Je složená ze splátky úroku a jistiny. Po smluvenou dobu se její výše nemění, mění se jen poměr mezi úrokem a splátkou jistiny. Na začátku splácení bývá vyšší úrok, ke konci doby splatnosti se úrok snižuje

Výdaj výloha či útrata je úbytek peněžních prostředků a peněžních ekvivalentů

Výnos je souhrnem peněžních prostředků, které podnik získá ze všech svých činností (produkce výrobků nebo služeb) za určité období

Volatilita je pravděpodobnost, že se cena, výnos, zisk atd. budou během určitého období měnit

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `app.jar` ..... spustitelný jar archiv s aplikací
- `app`
  - `src` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF