Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra systémového inženýrství

Diplomová práce

Vývoj a implementace doplňku pro řízení zásob do MS Excel

Bc. Vjačeslav Frasyňuk

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Vjačeslav Frasyňuk

Systémové inženýrství

Název práce

Vývoj a implementace doplňku pro řízení zásob do MS Excel

Název anglicky

Development and implementation of add-on for inventory managment for MS Excel

---

Cíle práce

Hlavním úkolem této práce je navrhnout a vyvinout doplněk pro aplikaci Microsoft Excel, který bude na základě vstupů uživatele řešit deterministické a stochastiké modely řízení zásob. Pro vývoj se použije přidružené prostředí Excelu pro psaní maker a aplikací v jazyce Visual Basic for Applications.

Metodika

Teoretická část práce vysvětluje teorii řízení zásob, základy statistiky nutné pro řešení stochastický modelů a základy algoritmizace a programování pro lepší pochopení problematiky.

Praktická část se podrobně zabývá převodem modelů řízení zásob do vývojových diagramů, na jejichž základech se pomocí Visual Basic for Applications naprogramuje doplněk pro Microsoft Excel. Bude bran ohled i na uživatele, kde se navrhne hierarchie modálních oken pro snadnější orientaci při výběru příslušného modelu. Funkčnost se ověří na některých úlohách a potom bude doplněk publikován pro používání ve výuce nebo praxi.

Doporučený rozsah práce

50-60

Klíčová slova

operační výzkům, řízení zásob, dodávkový cyklus, deterministické modely, stochastické modely, Visual Basic fo Application, Excel

---

Doporučené zdroje informací

EMMETT, S. *Řízení zásob : jak minimalizovat náklady a maximalizovat hodnotu.* Brno: Computer Press, 2008. ISBN 978-80-251-1828-3.

HILLIER, F S. – LIEBERMAN, G J. *Introduction to operations research.* Burr Ridge: McGraw-Hill Higher Education, 2005. ISBN 0-07-321114-1.

HUŠEK, R. – MAŇAS, M. *Matematické modely v ekonomii.* Praha: SNTL – Nakladatelství technické literatury, 1989. ISBN 80-03-00098-.

JABLONSKÝ, J. *Operační výzkum : kvantitativní modely pro ekonomické rozhodování.* Praha: Professional Publishing, 2007. ISBN 978-80-86946-44-3.

WALKENBACH, J. *Excel 2013 power programming with VBA.* Hoboken: John Wiley & Sons, 2013. ISBN 978-1-118-49039-6.

WRÓBLEWSKI, P. *Algoritmy : datové struktury a programovací techniky.* Brno: Computer Press, 2004. ISBN 80-251-0343-9.

Předběžný termín obhajoby

2018/19 ZS – PEF (únor 2019)

Vedoucí práce

Ing. Robert Hlavatý, Ph.D.

Garantující pracoviště

Katedra systémového inženýrství

Elektronicky schváleno dne 21. 2. 2018

doc. Ing. Tomáš Šubrt, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 22. 2. 2018

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 29. 03. 2019

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Vývoj a implementace doplňku pro řízení zásob do MS Excel" jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor(ka) uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 30.11.2016 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Robertu Hlavatému, Ph.D. za vstřícnost, trpělivost, ochotu a vedení mé diplomové práce. Dále bych rád poděkoval kolegům z práce za rady ohledně vývoje v prostředí VBA, rovněž své rodině a přátelům, kteří mě motivovali k dokončení studií na VŠ a měli pochopení, když jsem se jim nemohl věnovat.

Vývoj a implementace doplňku pro řízení zásob do MS Excel

Abstrakt

Tato diplomová práce navazuje na bakalářskou práci autora, která se se zabývala operačním výzkumem, přesněji metodami a modely řízení zásob. Původní práce spočívala ve statistické analýze dat a v aplikaci vybraných modelů pro optimalizaci dané situace. Ve vývojářském prostředí aplikace MS Excel se pomocí jazyku Visual Basic for Applications vypracovalo řešení, které umožňuje uživateli vyřešit vybrané modely řízení zásob bez nutnosti pamatovat si postupy, stačí pouze vstupní hodnoty. Teoretická část práce se podrobně věnuje jednotlivým metodám řízení zásob, prostředí MS Excel, jazyku VBA, metodám programování a návrhu softwaru. Praktická část popisuje přístup k tvorbě, samotné rozšíření, návod k jeho použití, ověření správné funkčnosti a možnosti, jak případně na tuto práci navázat.

Klíčová slova: řízení zásob, Excel, Visual Basic for Applications, deterministické modely, stochastické modely

Development and implementation of add-on for inventory management for MS Excel

Abstract

This diploma work follows the bachelor work of the author that was dealing with operational research, more precisely with methods and models of inventory management. Original work used statistical data analysis and applied selected models to optimize situation. In the developer environment of MS Excel, a solution was developed using Visual Basic for Applications which allows the user to solve selected inventory management models without having to remember procedures, data input is enough. The theoretical part deals with individual inventory management methods, MS Excel, VBA language, programming and software designing methods. The practical part of work describes the approach to developing of extension, extension itself, instructions of use, verification of the functionality and the possibilities how to follow up on this work.

Keywords: inventory management, Excel, Visual Basic for Applications, deterministic models, stochastic models

Obsah

Seznam obrázků

Seznam použitých zkratek

VBA – Visual Basic for Applications

MS – Microsoft

EOQ – Economic Order Quantity

NC – celkové náklady

UI – user interface

1 Úvod

Svět dnešních technologií se neúprosně rozvíjí víc a víc dopředu. To, co se dříve muselo počítat pouze „v ruce“ zdlouhavé hodiny, ne-li dny, se díky nástupu sálových počítačů dalo vyřešit za pár minut. S příchodem chytrých kalkulaček to už byla otázka zadaní hodnot a vztahů. A v dnešní době výkonných počítačů nám leckdy stačí zadat nebo naklikat potřebné hodnoty do programu a výsledek se nám počítá v nanosekundách.

Práce, kterou zrovna držíte v rukou, navazuje na dřívější bakalářskou práci autora a zabývá se metodami řízení zásob doplněné o vývoj nástroje, který umožňuje získat výsledky daných metod již po zadání vstupních hodnot. Nástroj vyžaduje tabulkový procesor Excel od společnosti Microsoft a vyvinutý je ve vývojářském prostředí aplikace v jazyce VBA. Od výsledku práce se očekává, že bude podávat správné výstupy daných metod a poslouží nejen jako pomůcka pro studenty operačního výzkumu.

V první části práce se nachází úvod do oboru operačního výzkumu, podrobněji rozebrané metody řízení zásob, popis vývojářského prostředí a přístupy k vývoji softwaru. Praktická část práce spočívá v programování pomůcky, proto se druhá část práce věnuje popisu tvorby, návodu k použití a zahrnuje i části zdrojového kódu pro ty, kteří chtějí vidět trochu více do hloubky nebo inspirovat se pro svou vlastní práci. V závěru se hodnotí funkčnost doplňku a komentuje možnost dalšího rozvoje.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním výstupem práce je studijní pomůcka, která na základě vstupů uživatele poskytuje vyřešené výsledky modelů řízení zásob. Doplněk je realizovaný v jazyku VBA (Visual Basic for Applications) a pro svůj chod vyžaduje tabulkový procesor Excel od společnosti Microsoft. Práce rovněž provede čtenáře jednotlivými metodami a poučí, jak s doplňkem zacházet. Výstup práce obsahuje počítadlo pro 3 deterministické a 3 stochastické metody, proto se na tuto práci dá dále navázat a rozšířit ji o další metody nebo jazykové lokalizace.

2.2 Metodika

K tvorbě a použití daného počítadla je zapotřebí být obeznámen s jednou z oblastí operačního výzkumu, a to s metodami řízení zásob. Tomuto tématu se věnuje první část diplomové práce, kde je podrobně rozebráno tři metody deterministické (bez povoleného nedostatku, s povoleným nedostatkem, produkčně-spotřební) a tři stochastické (s marginálním přístupem, s určením úrovně obsluhy, jednorázová zásoba). V závěru rešerše se nacházejí stručné znalosti a informace o vývoji softwaru a o prostředí MS Excel.

Druhá část spočívá v krátké analýze problému, volbě přístupu a samotné tvorbě učební pomůcky postavené na teoretické části. K programování je využito integrované vývojářské prostředí aplikace Excel a jazyk VBA, ve kterém si aplikace ukládá uživatelská makra a nastavení prostředí. Aby uživatelské prostředí bylo přívětivější, pro jednotlivé metody se vytvoří modální okna aplikace, kam se budou zadávat vstupní hodnoty.

V závěru práce se na vzorových příkladech ověří funkčnost doplňku a zhodnotí jeho přínosy a možnosti dalšího rozšiřování.

3 Teoretická východiska

V teoretické části práce se vyskytují znalosti nezbytné k úspěšnému nalezení cíle práce. V úvodu je stručně specifikován obor operačního výzkumu a dále jedna z jeho konkrétních disciplín, ze které táto práce vychází. Nezbytnou součástí je i problematika spojená s nástrojem vývoje, obecnou teorií programování a přístupy k řešení problému.

3.1 Operační výzkum

Operační výzkum. Pro spoustu lidí neznámý pojem, přitom se jedná o obor, se kterým se potkáváme na denní bázi. Jablonský (2007, s.9) uvádí, že počátky operačního výzkumu sahají už do meziválečného období minulého století a hlavní průkopníci, jako byl například G. B. Danzig nebo L. Kantorovič, se stali nositeli Nobelové ceny za ekonomii. Hlavním přínosem oboru je to, že pomocí matematicko-analytických metod pomáhá řešit mnoho situací. Pokud budeme vycházet z názvu, tak provádíme výzkum nad operacemi nebo aktivitami v rámci nějakého systému. Výzkum je způsob, kterým přistupujeme k řešení problému (Hillier a Lieberman 2001). Největší rozmach nastal v období druhé světové války a hned po její konci. S rostoucími nároky bylo zapotřebí vyvíjet nové postupy.

Nejznámějším a možná nejdůležitějším přínosem byla simplexová metoda, kterou vytvořil G. Danzig v roce 1947 pro řešení lineárních úloh. Rovněž se za války značně rozvinula i logistika a plánování. V rozvoji pak velmi pomohl příchod počítačové techniky, a to v té době nepředstavovalo elektronické zařízení, co se dnes vejde i do kapsy, ale obrovské halové mechanické počítače, jejichž vstupem a výstupem zpočátku bylo médium v podobě kartonového štítku s dírkami. To vydrželo téměř do 80. let minulého století, kdy světlo světa spatřily logické obvody v podobě elektronek, později jako mikroprocesory a čipy. A jak strmě se elektronika rozvíjela a s ní i výpočetní výkon, dařilo se realizovat nové postupy, provádět výpočty v řadech minut nebo dokonce provádět simulace chování systému v reálném čase.

3.1.1 Členění operačního výzkumu

Operační výzkum zahrnuje značné množství velmi různorodých modelů, které se zabývají rozdílnými oblastmi nejen ekonomického odvětví. Proto se obor s časem rozčlenil do relativně samostatných disciplín (Jablonský 2007). Například již zmíněná simplexová metoda patří do lineárního programování, jehož primárním cílem je řešení optimalizačních

úloh. Princip spočívá v nalezení extrému zvoleného kritéria, jež je definováno kriteriální funkcí o n proměnných a nachází se na množině možných variant, kterou vymezíme soustavou omezujících podmínek. Ty většinou jsou ve tvaru lineární či nelineárních rovnic nebo nerovnic.

Jednou z dalších disciplín je vícekriteriální rozhodování. Jedná se o poměrně mladou disciplínu a nevědomky jí používáme od doby, co začneme samostatně myslet. Zkoumá rozhodovací problémy, kde volíme mezi více variantami a které posuzujeme podle nějakých kritérií. Obzvláště oblíbená v managementu, kde se například musí zvolit vítěz výběrového řízení.

Za zmínku určitě ještě stojí teorie grafů, jež je hojně využívaná oborem řízení projektů. Grafy se zde rozumí síť tvořená uzly a hranami, které dané uzly spojují. V grafu můžeme hledat například nejkratší (kritickou) cestu. Teorie her je velmi podobná vícekriteriálnímu rozhodování. Liší se však v množství účastníků, kterých je víc jak jeden a musí volit strategie, která jim přinese zisk nebo nejmenší ztrátu. V úvodu ještě se zmiňovalo o simulacích, které slouží hlavně k analýze komplikovaných systémů a kde občas je jediným nástrojem výzkumu. Dle Jablonského (2007) je to však spíše prostředek analýzy než samostatná disciplína. Čemu se však z oboru operačního výzkumu věnuje táto práce je disciplína zvaná teorie řízení zásob.

3.2 Teorie řízení zásob

Co se rozumí pod pojmem zásoba? A co pod pojmem řízení zásob? Nejlépe se to vysvětlí na životní situaci, kdy například nakupujeme v obchodě a zrovna nějaká věc ze seznamu není dostupná. Na dotaz, zda to nemají na skladu uslyšíme, že bohužel došlo nebo nedorazilo. Kde tedy nastal problém? Na to se nám hodí právě modely řízení zásob, které tyto situace se snaží optimalizovat. Tato disciplína má za úkol zodpovědět otázky „kdy objednat/vyrobit?“ a „v jakém množství?“. U toho se ale musí hledět na výhodný poměr mezi náklady na skladování, objednání a ztrátu z nedostatku.

Když vytváříme zásobu, bereme v úvahu dobu mezi objednáním a dodáním zboží, kdy nám musí vystačit zbytek zásob na skladu. Zdánlivě existuje více přístupů, které však řeší stejný problém. Dle Emmeta (2008, s.43) sklad nám slouží jako jakási pojistka v nejistotě vůči dodavatelům, pokrývá neočekávánou poptávku a rovněž může sloužit jako fyzická ochrana, pokud ho považujeme za skladový objekt. Samotné skladování je nedílnou součástí

logistického systému, spojuje totiž výrobce (dodavatelé) a zákazníky, jinými slovy je na pomezí mezi místem vzniku zboží a místem jeho spotřeby. Pro vedení společnosti rovněž poskytuje důležité informace jako je stav zboží, podmínky nebo například rozmístění. Pomocí skladů můžeme překlenout prostor a čas, což nám pomůže se spokojeností zákazníků z jedné strany a se spokojeností výrobců z druhé strany, protože můžou plynule vyrábět dál.

Podíváme-li se do historie, většinu výrobků si vyráběl a prodával řemeslník sám. Nyní převážně platí, že není v silách podniku uživit se jediným druhem produktu, proto jich používá, přeměňuje, distribuuje nebo prodává vícero. Řízení zásob se výrazně podílí na financích, produkci a marketingu společnosti. Čím významnějším se stává nějaký produkt, tím důležitější je ho správně řídit (Wee 2011).

Z logického pohledu vyplývá, že pokud budeme tvořit moc velkou zásobu, bude v ní vázáno mnoho finančních prostředků, jež se můžou využít v jiném sektoru. A to nemusíme vycházet ani z teorie, aby nám došlo, že zboží má jistý životní cyklus a má náchylnost k degradaci, poškození nebo jinou formu ztráty hodnoty. Když ale zásoba bude nedostatečná, ztráta z nedostatku může být pro podnik likvidační. Nejenže negativně ovlivní vztahy se zákazníky, ale může odradit budoucí a poslat je ke konkurenci. Pokud dané zboží nevyrábíme, musíme si ho objednávat, což při časté frekvenci může být drahé. A tak hledáme jakousi rovnováhu, která se snaží minimalizovat rozdíly mezi vyjmenovanými veličinami a z nich plynoucí celkové náklady (Tomek a Hofman 1999).

3.2.1 **Používané proměnné**

V modelech počítáme s veličinami, které lze nebo nelze řídit. A jejich jediný rozdíl spočívá v tom, zda je management umí nebo neumí ovlivnit. Některé modely mají ještě pomocné veličiny, které jsou specifické pouze pro něj.

Mezi řiditelné proměnné zařazujeme:

- ***Velikost objednávky (Q)*** – určuje nám v jakém množství budeme obnovovat zásobu. Uvádí se většinou v jednotkách zboží/výrobku (kusy) nebo například v jednotkách hmotnosti, objemu aj.
- ***Úroveň objednávky (R)*** – představuje hladinu zboží na skladu v příslušných jednotkách, kdy provedeme objednávku. Někteří autoři označují jako bod znovuobjednávky.

- ***Délka dodávkového cyklu (t_c)*** – je rozdíl v časové linii mezi dvěma po sobě jdoucími objednávkami. Jednotky jsou většinou vedené ve dnech nebo měsících.
- ***Pojistná zásoba (w)*** – je součástí úrovně objednávky a slouží jako pojistka pro pokrytí nečekané spotřeby nebo poptávky.

Proměnné, které nemůžeme řídit:

- ***Celková roční poptávka (P)*** – charakterizuje očekávánou spotřebu zásoby za rok a její velikost je určovaná většinou specialistou na základě statistických údajů.
- ***Pořizovací lhůta dodávky (t_d)*** – je časový úsek mezi zadáním objednávky, přes reakci dodavatele, její přemístění a uskladnění u objednavatele.
- ***Jednotkové skladovací náklady (k_s)*** – je rozpočítaný náklad na uskladnění jedné jednotky zboží po dobu jednoho roku. Většinou zahrnuje náklad na pronájem nebo údržbu skladu, ztrátu hodnoty zboží, ztrátu z poškození, krádeže atd. Dle Jablonského (2007, s. 209-211) nemusí se jednat o konkrétní částku vyjádřenou v peněžních jednotkách, ale o procentuální podíl z ceny zboží.
- ***Jednotkové fixní pořizovací náklady (k_o)*** – je souhrnným nákladem za dopravu, manipulaci, administrativu a dalších nákladů souvisejících se zbožím. Již z názvu plyne, že se jedná o fixní částku nezávislou na velikosti objednávky. V praxi se můžeme setkat i možností variability proměnné, například když volíme mezi druhem transportu, ale pro zjednodušení budeme pracovat s konečnou pevnou částkou.
- ***Jednotkové náklady z nedostatku zásoby (k_n)*** – je variabilní proměnná, která závisí na velikosti neuspokojení poptávky a ztrátě z jedné jednotky. V praxi to znamená nejčastěji ušlý zisk, ztrátu z nedostatku zásoby, omezení prodeje nebo jiné. Někteří do toho počítají i špatnou pověst, pokud to dokážou kvantifikovat (Dömeová a Beránková 2004).

Pomocné a odvozené proměnné pro vyjádření některých ukazatelů, zejména celkových ročních nákladů:

- ***Celkové roční skladovací náklady (c_s)*** – představují veškeré náklady na celoroční skladování zásoby. Vzheledem k tomu, že zásobu vyskladňujeme a naskladňujeme, počítáme s průměrným stavem zásob za sledované období. Střední hodnotu stavu zásoby zjistíme buď statisticky nebo podle velikosti objednávky. Pro vyjádření nákladů používáme vztah:

$$c_s = \frac{Q}{2} \,.\, k_s$$

- ***Celkové roční (fixní) pořizovací náklady (c_o)*** – představují souhrn všech ročních nákladů vynaložených na realizované objednávky. Podílem celkové poptávky a velikosti objednávky získáme počet objednávek a vynásobíme ho nákladem na jednu objednávku:

$$c_o = \frac{P}{Q} \,.\, k_o$$

- ***Celkové roční náklady z nedostatku (c_n)*** – závisí na použitém modelu. Odvíjí se od míry neuspokojení poptávky a také od toho, jak dlouho ji neuspokojíme.
- ***Celkové roční náklady (NC)*** – jedná se o součet předchozích tří nákladů a nejčastěji je vyjadřujeme v peněžních jednotkách.

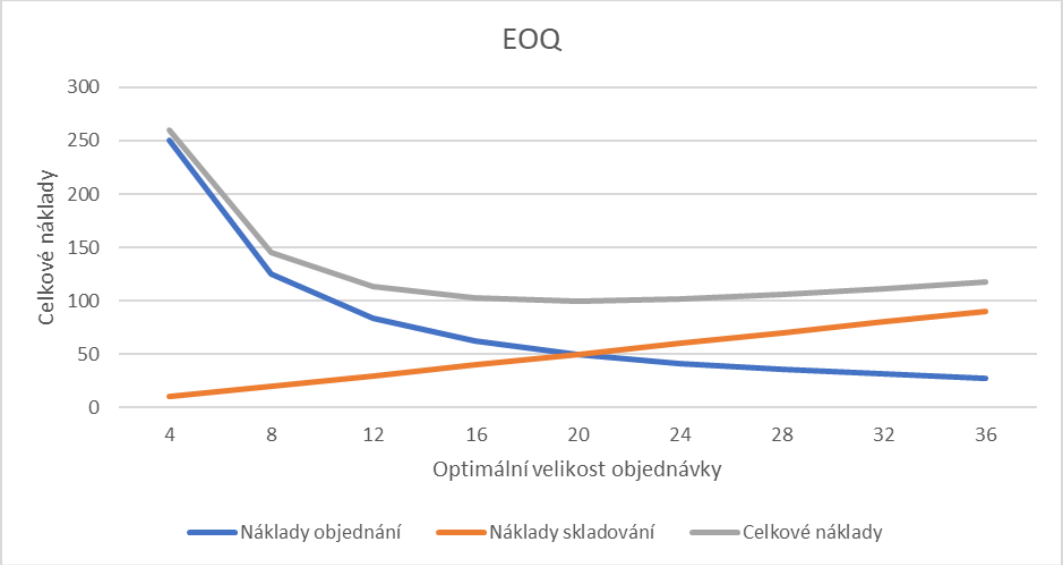
$$NC = c_s + c_o + c_n$$

3.2.2 Základní rozdělení modelů

Primárním hlediskem u modelů je celková (roční) poptávka, která může mýt charakter deterministický nebo stochastický. **Deterministická** poptávka v praxi znamená, že si jí pevně stanovíme na začátku nebo náš koncový zákazník si řekne o přesné množství. Ve výrobě se zas jedná například o polotovary, kde objem výroby je předem daný. **Stochastická** naproti tomu vychází ze statistiky a pravděpodobností, je tedy neurčitá. U nového zboží pracujeme pouze s pravděpodobností.

3.2.3 **Deterministické metody**

Jednou z nejběžnějších situací, se kterými se můžeme setkat je to, že spotřebováváme rovnoměrně, obzvlášť ve výrobním procesu, a následně doplňujeme zásoby novou dodávkou jednotek (Hillier a Lieberman 2001). I když se jedná o jeden z nejstarších modelů, používá se do dnes. Řeč je o modelu **EOQ** (economic order quantity), který nám jako výstup poskytne optimální velikost objednávky s ohledem na cenu skladování a cenu dopravy. V některých literaturách se můžeme setkat se zkratkou FOQ (fixed order quantity). U tohoto modelu nepřipouštíme nedostatek zásoby.



Obrázek 1: poměr nákladů modelu EOQ (zdroj: vlastní tvorba)

Na obrázku výše je znázorněn vztah nákladů na objednání a nákladů na skladování. Vodorovná osa nám určuje optimální velikost objednávky Q a svislá osa celkové náklady. Když budeme skladovat jen minimální množství, budeme muset častěji objednávat. Na druhou stranu, když budeme objednávat jen jednou za čas, budeme mít plný sklad, který se musí provozovat. Model nám pomůže najít správný poměr nákladů, kdy celkové náklady za objednávání a skladování budou dohromady nejnižší. V tomto bodě se nachází i optimální množství jedné objednávky. Pro odvození vzorce pro jeho výpočet existuje více způsobů. Ve způsobu zobrazeným níže je hledán extrém funkce celkových nákladů pomocí derivace.

$$NC = c_s + c_o = \frac{Q}{2}.k_s + \frac{P}{Q}.k_o$$

Nyní provedeme první derivaci pro Q:

$$\frac{dNC}{dQ} = \frac{k_s}{2} - \frac{k_o P}{Q^2} = 0$$

Výsledek položíme rovno nule a provedeme odmocnění, abychom ze vztahu dostali Q v první mocnině:

$$Q = \sqrt{\frac{2Pk_o}{k_s}}$$

Výsledný vztah můžeme dosadit do nákladové funkce a potom provést usměrnění zlomků, načež nám vznikne tato jednoduchá rovnice:

$$NC = \sqrt{2Pk_s k_o}$$

Když už víme, kolik objednávat a kolik nás to celkem bude stát, patří se ještě určit, kdy máme objednávat, aby zboží dorazilo/vyrobilo včas. Na to nám poslouží jednoduchý vztah, kde optimální velikost objednávky vydělíme celkovou roční poptávkou a vynásobíme jí 12. Získáme tak hodnotu vyjádřenou rovnou v měsících.

$$t_c = \frac{Q}{P} . 12$$

V praxi se bohužel nestává, že by se zboží po jeho spotřebě ihned doplnilo, pokud se nejedná o systém JIT (just in time). Nějakou dobu totiž trvá připravit a přepravit ho od výrobce, a proto máme takzvanou pořizovací lhůtu (t_d). Pomocí této proměnné lze zjistit bod znovuobjednávky (R) ze vztahu:

$$R = P.t_d$$

Důležité je nezapomenout převést jednotky pořizovací lhůty na roky. To znamená pro dny dělit 365 a pro měsíce dělit 12.

Na předchozím obrázku se nám nákladové funkce protínají v bodě, kde Q je rovno 20 a NC rovno 100. Pro demonstraci uděláme pár výpočtů a okomentujeme je. Vstupem modelu byla poptávka o velikosti 100, jednotkové skladovací náklady rovné 5 a jednotkové objednací náklady rovné 10. Dodací lhůta zboží byla stanovena na 1 měsíc. Pokud to pro kontrolu dosadíme do výše zmíněného vztahu, dostaneme:

$$Q = \sqrt{\frac{2Pk_o}{k_s}} = \sqrt{\frac{2.100.10}{5}} = \sqrt{400} = 20$$

A pro celkové náklady platí:

$$NC = c_s + c_o = \frac{Q}{2}.k_s + \frac{P}{Q}.k_o = \frac{20}{2}.5 + \frac{100}{20}.10 = 50 + 50 = 100$$

Nebo když použijeme jednodušší tvar, jehož výsledek je samozřejmě stejný:

$$NC = \sqrt{2Pk_sk_o} = \sqrt{2\,.100\,.5\,.10} = \sqrt{10000} = 100$$

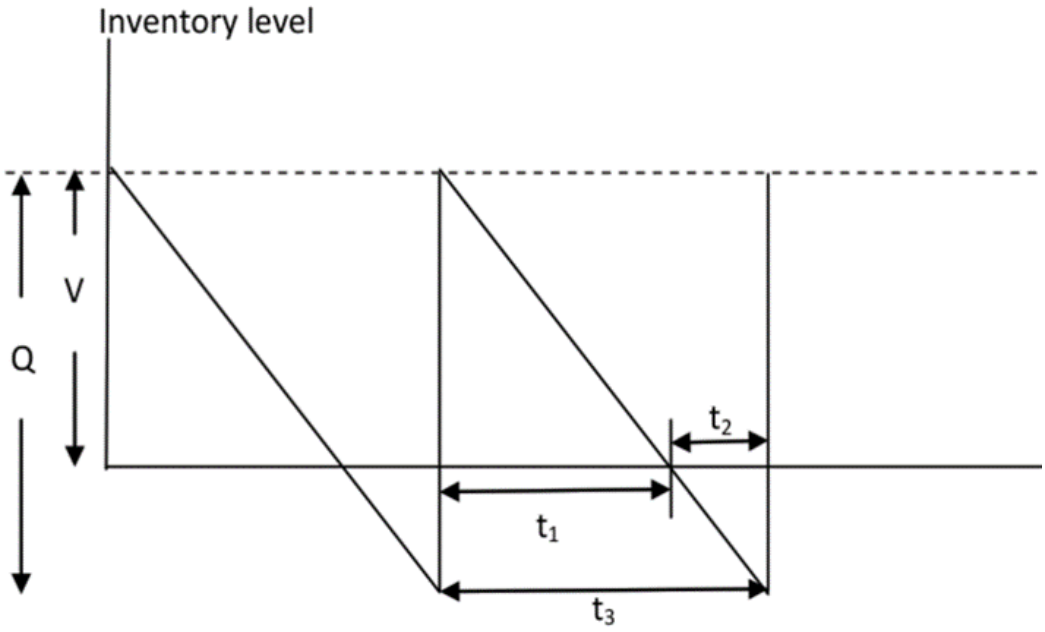
Z podílu celkové poptávky a optimální velikosti objednávky zjistíme, že budeme objednávat 5krát do roka, tzn. interval objednávky je 2,4 měsíce. Vzhledem k tomu, že dodání zboží bude trvat 1 měsíc, tj. 1/12 roku, budeme objednávat s předstihem, když zásoba ve skladu klesne na hodnotu:

$$R = P.t_d = 100\,.\frac{1}{12} = 8,\overline{33}$$

Když tedy předchozí výsledky okomentujeme dohromady, tak budeme objednávat 20 kusů zboží, 5krát do roka, a to pokaždé, když zásoba skladu klesne na hladinu 8,33 kusů. To celé nás bude na nákladech stát 100 peněžních jednotek.

Předchozí model však neumožňoval nedostatek zásoby a s tím spojenou neuspokojenou poptávku. U té se musíme rozhodnou, jaký má charakter, zda jí odložíme nebo uspokojíme z další objednávky. Pokud přistoupíme na první způsob s odložením, rozdělí se dodávkový cyklus na dvě části, a to interval spotřeby (t_1) a interval nedostatku (t_2). Celkovou dobu dodávkového cyklu vyjádříme součtem těchto intervalů:

$$t_3 = t_1 + t_2$$



Obrázek 2: přechodné uspokojování poptávky (zdroj: Wee, Hui-Ming, 2011)

Náklady za skladování očistíme o průměrný nedostatek zásoby (s/2) a vynásobíme intervalem spotřeby (t_1) a náklady z nedostatku zásoby lze odvodit tak, že dáme do součinu průměrný nedostatek zásoby (s/2) s jednotkovými náklady z nedostatku (k_n) a intervalu nedostatku (t_2).

$$NC = c_s + c_o + c_n = \left(c_s \,.\, \frac{(Q-s)}{2} \,.\, t_1 + c_o + \, c_n \,.\, \frac{s}{2} \,.\, t_2 \right) .\, \frac{P}{Q}$$

Pro zjednodušení vztahu celkových nákladů rozepíšeme oba intervaly jako poměry optimální velikosti objednávky (Q) a nedostatečné zásoby (s), kde platí:

$$t = \frac{Q}{P}$$

$$\frac{t_1}{t} = \frac{Q-s}{Q} \quad \rightarrow \quad t_1 = \frac{Q-s}{Q} \cdot \frac{Q}{P} = \frac{Q-s}{P}$$

$$\frac{t_2}{t} = \frac{s}{Q} \quad \rightarrow \quad t_2 = \frac{s}{Q} \cdot \frac{Q}{P} = \frac{s}{P}$$

Funkce pro celkové náklady pak po dosazení a roznásobení má následující podobu:

$$NC = k_s \cdot \frac{(Q-s)^2}{2Q} \cdot t_1 + k_o \cdot \frac{P}{Q} + k_n \cdot \frac{s^2}{2Q} \cdot t_2$$

Pro zjištění hodnoty optimálního objednávaného množství, stejně jako v předchozím modelu, aplikujeme derivací podle Q a dostaneme:

$$Q = \sqrt{\frac{2Pk_o}{k_s}} \cdot \sqrt{\frac{k_s + k_n}{k_n}}$$

Jak je ze vztahu patrné, není závislý na délce nedostatku, ale pouze na poměru nákladů ze skladování a nedostatku. To samé platí i pro velikost nedostačující zásoby, kterou zjistíme zase derivací, tentokrát už podle proměnné s:

$$s = Q \cdot \frac{k_s}{k_s + k_n}$$

Když tyto vztahy dosadíme do rovnice celkových nákladů a stejně jako u předchozího modelu provedeme usměrnění zlomků, znovu dosáhneme elegantnějšího a jednoduššího vztahu, který má následující tvar:

$$NC = \sqrt{2Pk_s k_o} \,.\, \sqrt{\frac{k_n}{k_s + k_n}}$$

Jak si lze všimnout, vztah je stejný jako v předchozím modelu, ale přibyl k němu poměr nákladů na skladování a nedostatku. Vzhledem k tomu, že hodnota daného podílu v druhé odmocnině bude vždy menší 1, budou i celkové náklady nižší než v prvním modelu. Pokud budeme vycházet ze zadání předchozího příkladu, kde původní hodnoty doplníme o náklad z nedostatku roven 4, interval spotřeby 4 a interval nedostatku 2 dostaneme tyto výsledky:

$$Q = \sqrt{\frac{2Pk_o}{k_s}} \,.\, \sqrt{\frac{k_s + k_n}{k_n}} = 20 \,.\, \sqrt{\frac{5+4}{4}} = 20 \,.\, \sqrt{2{,}25} = 20 \,.\, 1{,}5 = 30$$

$$s = Q \,.\, \frac{k_s}{k_s + k_n} = 30 \,.\, \frac{5}{5+4} = 30 \,.\, \frac{5}{9} = 16, \overline{66}$$

$$NC = \sqrt{2Pk_s k_o} \,.\, \sqrt{\frac{k_n}{k_s + k_n}} = \sqrt{10000} \,.\, \sqrt{\frac{4}{9}} = 100 \,.\, \frac{2}{3} = 66, \overline{66}$$

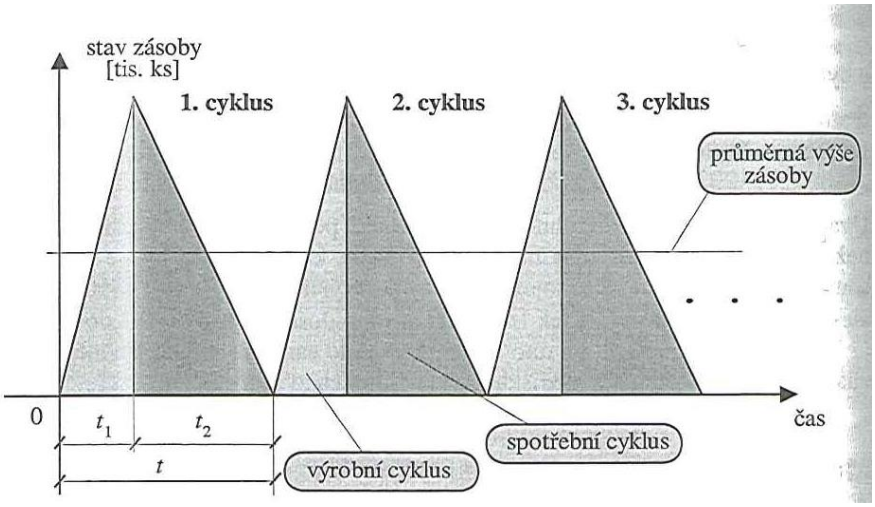
Nesmíme zapomenout na stanovení hodnoty cyklu dodávky a úrovně objednávky, kterou oproti minulému modelu ponížíme o neuspokojenou zásobu:

$$t_c = \frac{Q}{P} \,.\, 12 = \frac{30}{100} \,.\, 12 = 3{,}6$$

$$R = P \,.\, t_d \; - s = 100 \,.\, \frac{1}{12} - 16{,}66 = \; -8{,}33$$

Z těchto výsledků zjistíme, že budeme objednávat 30 kusů zboží, každých 3,6 měsíce pokaždé, když neuspokojíme poptávku 8,33 kusů. Celkem nás to bude stát 66,66 peněžních jednotek a v porovnání s prvním modelem nám to ušetří 33 % nákladů.

Posledním vybraným deterministickým modelem je produkční, který většinou označujeme zkratkou POQ (production order quantity) nebo jako produkčně-spotřební. Vychází ze stejných předpokladů jako první model, ale s tím rozdílem, že sklad nedoplňujeme jednorázově a dodávkový cyklus je složen z cyklu výroby a cyklu spotřeby. V prvním jmenovaném produkujeme a spotřebováváme zároveň, v druhém pouze spotřebováváme. Z toho plyne, že musíme dodržovat vyšší intenzitu produkce (pr) než spotřeby (p). Když se v druhém cyklu spotřebuje veškerá zásoba, startujeme novou výrobní dávku a neustále se to opakuje. Nepředpokládáme, že budeme mít nedostatek zásoby.



Obrázek 3: výrobní a spotřební cyklus (zdroj: Jablonský, 2007)

Tento model pracuje se skladovacími náklady (k_c) a fixními náklady na spuštění produkční dávky (k_o). Předběžná nákladová funkce pak má tento tvar:

$$NC = k_s \,.(průměrná\ zásoba) + k_o.(počet\ výrobních\ cyklů/rok)$$

Ze vztahu, kde platí, že maximální zásoba (Q) je určená délkou produkčního cyklu (t_1) a intenzitou produkce (pr), získáme průměrnou zásobu tím, že jí podělíme 2. Nesmíme ale zapomenout tuto hodnotu ještě vynásobit poměrem produkce a spotřeby, protože vyrábíme a spotřebováváme zároveň. Průměrná zásoba se pak zjistí tímto vztahem:

$$průměrná\ zásoba = \frac{pr - p}{pr} \,.\frac{Q}{2}$$

Počet výrobních cyklů se vyjádří podílem celkové poptávky a maximální zásoby. Po dosazení do nákladové funkce změní nám tvar na:

$$NC = k_s \,.\frac{pr - p}{pr}\,.\frac{Q}{2} + k_o.\frac{P}{Q}$$

V extrému této funkce se nachází hodnota optimální (maximální) zásoby Q, proto tuto funkci derivujeme podle Q a dostaneme tento vztah:

$$Q = \sqrt{\frac{2\,.P\,.k_o}{k_s}}\,.\sqrt{\frac{pr}{pr - p}}$$

Když tuto funkci dosadíme zpět do nákladové funkce a usměrníme zlomky, dostaneme tento vztah:

$$NC = \sqrt{2\,.P\,.k_o.k_s}\,.\sqrt{\frac{pr - p}{pr}}$$

Pro demonstraci výstupů zmíněných vztahů použijeme předchozí zadání, které doplníme o intenzitu produkce rovnou 20 ks/měs. a o intenzitu spotřeby rovnou 11 ks/měs. Příprava výrobní dávky trvá půl měsíce. Po dosazení do vzorečků zjistíme, že:

$$Q = \sqrt{\frac{2\,.P\,.k_o}{k_s}}\,.\sqrt{\frac{pr}{pr - p}} = 20\,.\sqrt{2{,}22} = 20\,.1{,}49 = 29{,}8$$

$$NC = \sqrt{2\,.P\,.k_o.k_s}\,.\sqrt{\frac{pr - p}{pr}} = 100\,.\sqrt{0{,}44} = 100\,.0{,}67 = 67$$

Vyrábět budeme do té doby, dokud nedosáhneme hladiny 29,8 kusů. Jednoduchými podíly zjistíme, že optimální délka intervalu produkce je 3,57 měsíce a že délka produkce je 1,49 měsíce. Rozdílem těchto hodnot získáme délku spotřebního cyklu 2,08 měsíce.

Maximální zásoba na skladu dosáhne hladiny 13,4 kusů. Tím, že příprava výroby zabere půl měsíce, určíme ještě bod znovuobjednávky:

$$R = P.t_d - s = 100 . \frac{1}{12} . \frac{1}{2} = 4{,}16$$

Pokud tedy nám zásoba na skladu klesne pod hladinu 4,16 kusů, zahájíme přípravu na další produkční cyklus.

Vedle těchto modelů existuje například model s množstevními slevami, kde náklad objednávky je závislý na objednaném množství. Model JIT (just in time) zas vyžaduje úzkou spolupráci mezi dodavatelem a odběratelem a spoléhá na vysokou frekvenci dodávek s nízkým nákladem objednávky (Dömeová a Beránková 2004). Bohužel tyto modely nebudou součástí doplňku, a proto se zde naskytuje možnost budoucího rozšíření.

3.2.4 **Stochastické metody**

Jak bylo uvedeno v úvodu kapitoly, stochastické modely pracují s pravděpodobnostmi a neurčitou velikostí poptávky. Hlavním cílem je zase minimalizace nákladů. Navíc nám zde přibývají pojmy z oboru statistiky, a to normální rozdělení, směrodatná odchylka apod. Tím, že poptávka není rovnoměrného charakteru, je velmi obtížné určit náklady z nepokrytí. Podle Jablonského (2007) mezi základní předpoklady těchto modelů řadíme:

- konstantní doba pořízení objednávky
- náklady z nedostatku nejsou ovlivněné dobou trvání nedostatku
- poptávka ve sledovaném období má normální rozdělení a je spojitá
- pojistná zásoba má kladnou hodnotu
- optimální objednací úroveň je vyšší než střední hodnota poptávky v objednací lhůtě

Objednávky pak vystavujeme, když zásoba klesne pod optimální objednací úroveň (R). Pro řešení problémů se stochastickou poptávkou používají se základní dva druhy modelů:

- s možností opakovat objednávku (znovuobjednávka)
- s jednorázovou objednávkou

V kategorii s opakovanou objednávkou existují dva velmi podobné modely, které se liší přístupem k hledání optimální objednací úrovně. Model s marginálním přístupem přidává marginální náklady, dokud se náklad na skladování pojistné zásoby se nevyrovná s nákladem spojeným s neuspokojením poptávky. Samotnou poptávku nevíme a místo ní používáme poptávku odhadovanou, kterou označujeme jako P s pruhem:

$$Q = \sqrt{\frac{2\bar{P}k_o}{k_s}}$$

Než dojdeme k hodnotě optimální úrovně R, dáme jí zpočátku rovnou střední hodnotě poptávky během pořizovací lhůty:

$$R = \bar{M}$$

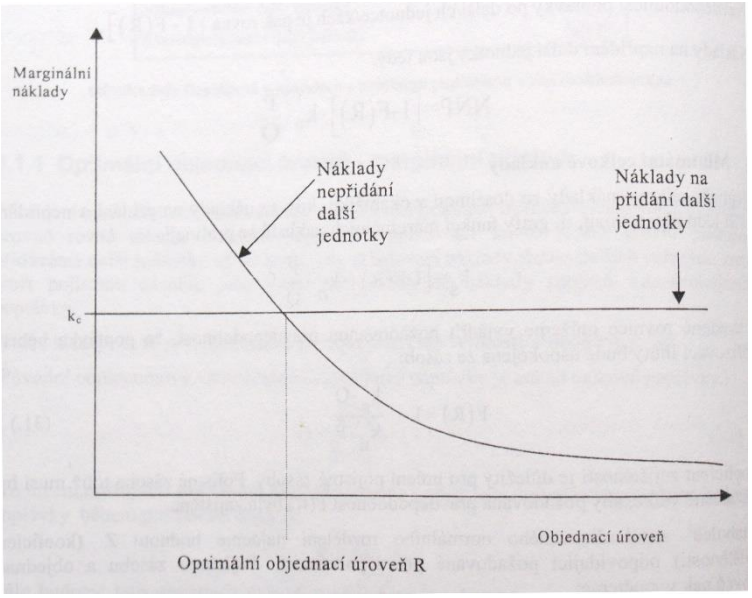
Zbývá nám už jen určit náklady na přidání (NP) či nepřidání další jednotky (NPP). Pokud jednotku přidáme, budeme muset jí mít na skladu, a proto náklad na přidání odpovídá přibližně nákladu za roční skladování jedné jednotky.

$$NP = k_s$$

Náklad na nepřidání je závislý na pravděpodobností, že během objednací lhůty poptávka nepřesáhne objednací úroveň a zvládneme ji uspokojit ze zásob. Zásobu musíme vynásobit nákladem na skladování a počtem objednávkových cyklů. Pravděpodobnost uspokojení definujeme jako F(R), proto pravděpodobnost neuspokojení je zbytek po odečtení od 1.

$$NPP = [1 - F(R)] \,.\, k_s \,.\, \frac{\bar{P}}{Q}$$

Zatímco náklady na přidání další jednotky jsou konstantní, náklady na nepřidání mají exponenciální charakter. Aby se docílilo minimální nákladů, musí se oba náklady rovnat, jak je to znázorněno na následujícím obrázku.



Obrázek 4: marginální náklady (zdroj: Dömeová, 2004)

Když dáme oba vztahy do rovnosti, vyjádříme neznámou pravděpodobnost F(R):

$$k_s = [1 - F(R)].k_s.\frac{\overline{P}}{Q}$$

$$F(R) = 1 - \frac{k_s \,.Q}{k_n \,.\overline{P}}$$

Dle statistických tabulek, konkrétně standardizovaného normálního rozdělení, můžeme pro výslednou pravděpodobnost dohledat koeficient zajištěnosti Z. Ten tam slouží k výpočtu pojistné zásoby (w):

$$w = Z.\sigma_M$$

Tuto hodnotu přičítáme ke stávající optimální objednávací úrovni a výsledná hodnota nám udává, při jakém množství zboží máme provést další objednávku:

$$R = \overline{M} + w$$

Pro stanovení celkových ročních nákladů NC je zapotřebí zahrnout všechny faktory z tohoto modelu. Obecně se funkce skládá ze dvou částí:

- nákladů na objednání a nákladů z neuspokojení poptávky ovlivněných pravděpodobností
- nákladů na skladování

$$NC_{(Q,R)} = \left[k_o + k_n.\sigma_M\,.N(Z)\,.\,\frac{\bar{P}}{Q}\right] + \left[\frac{Q}{2} + (R - \bar{M})\right].k_s$$

Stanovit optimální objednací úroveň můžeme ještě jedním způsobem, kde sami určíme úroveň obsluhy. Tou určíme, jak velkou část z celkové poptávky uspokojíme ze zásob. Vychází z předchozího modelu, kde má společnou část s výpočtem optimální úrovně objednávky. Ke stanovení objednací úrovně musíme určit množství neuspokojených poptávek, a to pomocí funkce pro koeficient zajištěnosti (Dömeová a Beránková 2004).

$$k' = \sigma_M\,.\,\tau(k)$$

Tento vztah rozšíříme o celkovou poptávku, aby se zjisti podíl neuspokojené:

$$\frac{\sigma_M\,.\,\tau(k)}{Q} = 1 - PP$$

Z této funkce můžeme vyjádřit samotnou pomocnou funkci τ(k):

$$\tau(k) = \frac{Q(1 - PP)}{\sigma_M}$$

Postup pokračuje tak, že dle tabulky s koeficienty zajištěnosti dohledáme konkretní hodnotu k a nalezenou hodnotou vynásobíme směrodatnou odchylku σ_M , jak tomu bylo v předchozím modelu. S rostoucí úrovní obsluhy nám téměř exponenciálně roste pojistná zásoba. Výsledný vztah pro stanovení úrovně objednávky pak vypadá takto:

$$R = \bar{M} + k\sigma_M$$

Posledním modelem, se kterým se setkáme v této práci je model s jednorázovou spojitou objednávkou. (Výraz spojitá zde není uveden nadarmo, protože existuje nespojitá poptávka, tvořená intervaly a pravděpodobnosti se musí kumulovat. Jedná se o další možné rozšíření doplňku.) Tento model je užitečný, když si vytvoříme zásobu jen jednou a nehodláme ji už doplňovat, případně s dodatečným nákladem (Jablonský 2007). Poptávka je stále stochastická, kde z předchozích zkušenosti, průzkumů nebo studií jsme schopni určit střední hodnotu a směrodatnou odchylku. V praxi se s tímto problémem setkáme například u sezonního zboží, jako je oblečení, vánoční stromky, pomlázky a dále třeba u zboží, které se kazí nebo rychle ztrácí na hodnotě, kam můžeme zařadit potraviny, květiny nebo tisk.

Když vytváříme počáteční zásobu Q, můžou nastat tři situace:

1. **skutečná poptávka P je nižší než naše zásoba Q –** část zboží nám zůstane na skladu a jeho zůstatková hodnota může být rovna nule (když se něco zkazí) + náklady na jeho pořízení a skladování
   c_1 = nákupní cena + dodatečné náklady – zůstatková hodnota
2. **skutečná poptávka P je vyšší než naše zásoba Q –** nemůžeme uspokojit celou poptávku, a tak nám vzniká náklad ušlého zisku
   c_2 = prodejní cena – nákupní cena – dodatečné náklady
3. **skutečná poptávka P je stejná jako zásoba Q –** nevznikají nám ani ztráty ani další náklady

Pracujeme pak s pravděpodobností, že nedojde k nedostatku zásoby a s tím spojenou neuspokojenou poptávku. To vyjádříme vztahem:

$$\gamma = \frac{c_2}{c_1 + c_2}$$

Pro výslednou pravděpodobnost v tabulkách normálního rozdělení najdeme odpovídající hodnotu z, výsledná počáteční zásoba se pak rovná:

$$Q = \overline{P} \,.\, z \,.\, \sigma_M$$

3.3 Úvod do programování a vývoj aplikací

Jak uvádí Hylmar (2009), programování je dnes jeden z nejdůležitějších oborů lidské činnosti. Jistou formou počítače (procesor, čip, server apod.) je dnes řízeno téměř vše: hodinky, pračky, požární systémy, mobilní telefony, vozidla nebo i vesmírné rakety. Aby nám daný počítač dělal, co po něm chceme, musíme ho nejdříve naprogramovat. Než se ale vrhneme detailněji k programování, je na místě zmínit něco z historie počítačů.

3.3.1 Historie počítačů

První stopy počítání nalezneme již u Sumerů, kteří měli vlastní počítací tabulky, kalendář a měrný systém. Ano, zatím to nedělá, co po tom chceme, ale na nějakých základech se budovat muselo. V Číně existovalo počítadlo abakus, které si můžeme pamatovat ze školek a škol jako kuličky na vodorovných tyčkách. Už se jednalo o systém, který nám usnadňoval počítání. Když přeskočíme vynálezy B. Pascala a G. W. Liebnize a přesuneme do začátku 19. století, narazím na tkalcovský stav J. M. Jacquarda, který už s programováním začíná souviset, protože tkal vzor pomocí děrovaných karet, které v té době představovali paměť v binární podobě. V 1833 C. Babbage částečně postavil stroj, který uměl počítat některé matematické funkce. Skládal se z úložiště, výpočetní jednotky a řídícího mechanizmu. Tímto vynálezem se nadchla Ada Lovelace a vytvořila pro něj první teoretické programy, proto se považuje za jednu z prvních programátorů v historii informatiky. Stroj byl kvůli mechanické náročnosti postaven až dvacet let poté v podobě prototypu. Nadšenci stroj do konečné podoby kompletně dostavěli až v 1992. Vraťme se ale zpátky, do 1890, kdy se objevili první stroje na děrné štítky. Tyto stroje zpracovávali statistická data při sčítání lidu a postavil ho H. Hollerith. Jeho podnik se v 1911 transformoval na společnost International Business Machines Corp. Jsou Vám iniciály společnosti nějak povědomé? Ano, jedná se skutečně o společnost, která existuje dodnes a známe jí pod zkratkou IBM. Ve třicátých letech vzniká teorie algoritmů, na které se podíleli jména jako Turing, Markow nebo Gödel. Známý je především Turingův abstraktní výpočetní stroj, jehož matematickém modelu posloužil jako základ pro konstrukci moderních počítačů. Ve 40. letech vznikají první univerzální počítače, obzvlášť tomu přispěl válečný konflikt ve světě. První kalkulátor Mark 1 pochází z roku 1944 a byl postaven na teorii C. Babbage a realizovaný elektromechanicky na Harvardové univerzitě. O dva roky později přišel první elektronický počítač ENIAC (Electronic Numerical Interpreter And Calculator)

z Pensylvanské univerzity a sloužil pro řešení balistických výpočtů. Obecně ale považujeme za první počítač v plném smyslu slova počítač EDVAC (Electronic Discrete Variable Automatic Computer) z Princetonské univerzity. Obsahoval kompletní program i výpočty ve společné paměti počítače. S touto myšlenkou přišel John von Neumann. Proto dnes rozlišujeme počítače na Harvardový přístup a von Neumanna. Po válce rozvoj techniky díky studené válce strmě stoupal vzhůru, přicházeli kalkulačky od IBM a vědecké sálové počítače. Nyní, dovolíme-li použít anglicismus, procházíme érou komputerizace, a to takřka ve všech oblastech života. Počítače máme všudypřítomné a neumíme si bez nich život představit.

3.3.2 **Programování a algoritmizace**

Programování není pouze o vytváření programu pro počítač, ale snažíme se o co nejefektivnější, nejrychlejší a nejjednodušší přístup k jeho tvorbě. Ano, počítač sice pochopí, co po něm chceme, ale proč mu to neříci srozumitelněji, aby s tím měl co nejméně práce? Před samotnou tvorbou je důležité si srovnat, co chceme, aby to dělalo a jak to budeme tvořit, abychom se vyvarovali plýtvání času nebo neustálého stavění od nuly. Wroblewski tuto etapou pojmenovává jako reflexi. Potom vzniká zdrojový kód, většinou jako textový soubor a pro psaní využívá integrované vývojové prostředí (IDE), které kombinuje všechny nástroje, jež jsou během programování potřeba. Některá prostředí obsahují i grafické editory, v nichž můžeme sestavit i uživatelské rozhraní (User Interface – UI) bez psaní jediné řádky kódu. Výsledkem programová je většinou skupina souborů, kde je v symbolické formě popsáno chování výsledného programu. To je zapsáno programovacím jazykem (většinou v angličtině) a pro jeho překlad pro stroj slouží kompilátor. Než zvolíme jazyk, kterým budeme s počítačem komunikovat, věnujme chvilku stavbě algoritmu.

Wroblewski (2004) považuje algoritmizaci / algoritmiku za vědní obor, který přinesl mnoho efektivních nástrojů, které umožňují řešit nejrůznější problémy pomocí počítačů. Každý autor definuje pojem algoritmus po svém. Wroblewski (2004) cituje definici ze slovníku *Le Nouveau Petit Robert*, kde algoritmus je konečnou posloupnost či sekvenci pravidel, která je aplikována na konečnou množinu dat a umožňuje řešit třídu problémů podobného typu nebo sada pravidel, která je typická pro jisté informatické výpočty nebo činnosti. I když definice jsou jasné a srozumitelné, jsou celkem obecné. Dle Prokopa (2012) algoritmus je konečná posloupnost kroků, po jejichž provedení dojdeme k určitému předem

vytčenému cíli. Musí splňovat následující vlastnosti: být konečný, tzn. skončit po konečném počtu kroků, jednotlivé kroky algoritmu musí být definovány jednoznačně, mýt jasné vstupy a výstupy a algoritmus musí skončit v „rozumně krátkém čase“, tzn. musí být efektivní.

Pro znázornění algoritmu existují dva přístupy, kde se snažíme přiblížit počítači nebo člověku. V prvním případě použijeme jazyk symbolický adres například v jazyce assembler. Ten ale není pro neznalého člověka srozumitelný, proto se volí většinou abstraktní přístup v podobě vývojových diagramů, které člověk s dostatečnou inteligencí je schopen „přelouskat“. Algoritmy v praktické části budou jednoduché podoby, kde bude pouze vstup, kontrola vstupu, výpočet a zapsání výsledku do nového listu.

3.4 Microsoft Excel a VBA

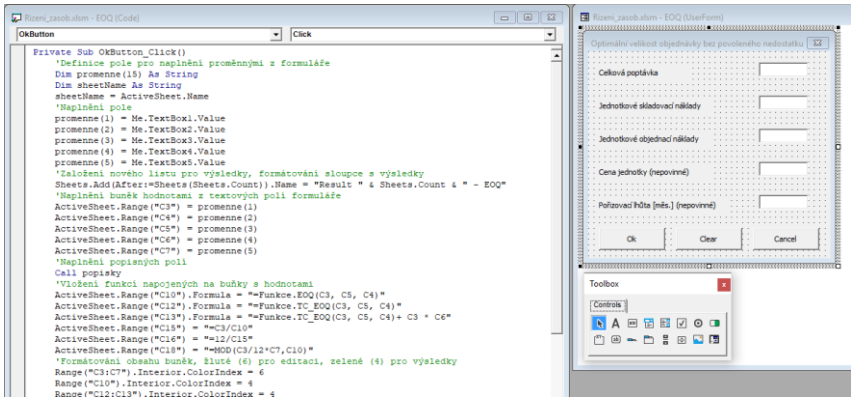
Dle Urtise (2011) je aplikace Excel nejmocnější a nejvíc používaný tabulkový procesor na světě. První verze spatřila světlo světa v roce 1987 a program byl a stále je vydáván nejen pro operační systém Windows, ale i na macOS, Android a iOS. Aktuální verze je Excel 2016, kde společnost dále plánuje po vzoru například společnosti Adobe, vést jednu verzi/službu, která se bude do budoucna rozšiřovat. Její současný název je Office 365 a zahrnuje nejen Excel, ale i mnoho další aplikací z balíčku MS Office a navíc je přístupná skrze webové rozhraní. Struktura programu je postavená na hierarchii objektů, kterých je více jak 200 a zahrnují pojmy jako je sešit, list, funkce, rozsah apod.

Mezi základní funkce aplikace patří práce s čísly, grafické výstupy, kontingenční tabulky a například nahrávání maker. Součástí vybraných aplikací je i zabudovaný IDE (Integrated Development Environment) s editorem jazyku VBA. Samotný jazyk je též dílem Microsoftu, vychází z událostmi řízeného jazyka Visual Basic 6, který už není podporován a poprvé se objevil už v roce 1993. Aktuální verze jazyku VBA je již 7.1 s podporou 64-bitových OS a kromě tvorby vlastních funkcí nám umožňuje automatizovaní pomocí maker (pouze v dané aplikaci), přistupovat k Windows API nebo k dalším funkcím pomocí dynamicky linkovaných knihoven DLL. Samotnou strukturu jazyka a všechny jeho objekty autor záměrně nepopisuje, protože na toto téma je napsáno nespočet „kuchařek“ o rozsahu stovkek stránek textu. V práci se to omezí pouze na komentář kódu. Jazyk je objektově orientovaný a jednotlivé objekty mají vlastní skupinu vlastností a metod (Urtis, 2011). Martin (2012) pro změnu tvrdí, že tom zas tak úplně není. Nicméně poskytuje nám to spoustu možností v programování s tím, jak kód uzavřeme (schováme) do třídy.

K editoru VBA se v aplikaci Excel dostaneme buď skrz povolenou skrytou záložku „Vývojář“ nebo pomocí klávesové zkratky Alt+F11. Zpravidla platí, že když něco nedokážeme v aplikaci udělat standardními postupy, je šance, že se nám to podaří pomocí VBA. Do podrobná každá funkce popsaná nebude, ale ve zkratce se dá projekt rozdělit na několik sekcí/okruhů:

- Objekty Excelu
- Formuláře (forms)
- Moduly
- Třídy

O objektech jsme se zmínili v úvodu a lze mezi ně zařadit sešit, list atd. Na co už ovšem běžně nevidíme jsou formuláře, do kterých zadáváme vstupy. Mají podobu grafickou, kterou můžeme sami dle možností navrhnout, a na ní navázanou podobu kódovou, kde na jednotlivá tlačítka a textová pole navážeme funkce a další akce, které se mají vykonat.



Obrázek 5: formuláře ve VBA editoru (zdroj: vlastní tvorba)

Další sekcí jsou moduly, do kterých zadáváme funkční kód. Teoreticky lze to samé zadávat již do formulářů, ale pro korektnost a správné fungování funkce programujeme zde a z formulářů je pomocí procedur voláme.

Posledním oddílem je sekce tříd. Pomocí tříd můžeme naprogramovat i vlastní objekt aplikace. K objektům můžeme vypracovat jejich vlastní procedury a akce, přidat jejich vlastnosti apod. Výsledek už není vázaný na soubor, jak tomu je u formulářů a modulů, ale rovnou na aplikaci Excel. Pro přenos mezi aplikacemi by se musel vygenerovat doplněk ve formátu Add-In a manuálně instalovat v každé instanci, pokud se jedná o prostředí školy. Pro účely této práce by se samozřejmě daly použít, na druhou stranu by to vyžadovalo více stráveného času a pro sestavení modelů řízení zásob až zbytečně složitý postup.

3.4.1 Návrh uživatelského prostředí

Určitě se v praxi každý setkal s aplikacemi, které nemají zrovna přívětivé prostředí. Uživatel se pak potýká s problémy najít potřebnou ikonu nebo pro realizaci musí provádět „krkolomné“ úkony. Dnes již běžná věc je rozdělovat prostředí na front-end a back-end. Jak tvrdí Albright (2012), práce programátora spočívá ve vytvoření user-friendly aplikace s modelem (nejlépe skrytým před uživatelem), který je obklopený právě front-endem a back-endem. Front end zde prezentuje se jako dialogové okno nebo formulář, který umožní uživateli definovat problém. Po zadání vstupu aplikace převezme informace, vybuduje správný model, pokud je potřeba tak i optimalizuje a eventuálně prezentuje výsledky zpět uživateli, nejlépe doplněné o grafy. Existuje mnoho přístupů, jak designovat uživatelské prostředí, aby se v něm každý vyznal i bez nápovědy. Například při vytváření vstupních formulářů Král (2010) doporučuje se držet těchto pravidel:

1) Užití bez velmi složité nápovědy
2) Intuitivní používání prvků
3) Počet aktivních prvků co nejnižší
4) Jasná formulovatelnost
5) Logicky rozdělovat do skupin
6) Když je proces složitý, rozdělit ho do více jednoduchých formulářů

Dalším z doporučení Krále je (2010) je držet kód a formuláře odděleně, omezit se jen na to nejnutnější a potřebné funkce volat z modulů. Autor se s doporučením ztotožňuje, avšak u „jednoduššího“ projektu je to odpustitelné a alespoň se snaží dělat kompromis, kde důležité funkce jsou volány z modulu funkcí, zatímco vkládání a formátování hodnot je realizováno kódem formuláře.

Při návrhu je vhodné neopomenout funkce, které nám umožní formulář zavřít nebo skrýt. Bylo by to dost nepohodlné pro uživatele, který je na tyto „standardizované“ postupy zvyklý z ostatních aplikací. V základu jsou sice modální okna „vybaveny“ ikonkou křížku pro uzavření, ale proč to neusnadnit a po akci formulář nezavřít automaticky? Po zadání dat a stisknuti tlačítka „Ok“ se zpracuje kód, který je danému tlačítku přidělen. Když ale na konec funkce přidáme příkaz *Unload Me*, najednou je život jednodušší. Formulář zmizí a nám zůstane čistá pracovní plocha. To samé platí i pro tlačítko „Cancel“, které ovšem po skrytí zavolá zpět hlavní menu.

4 Vlastní práce

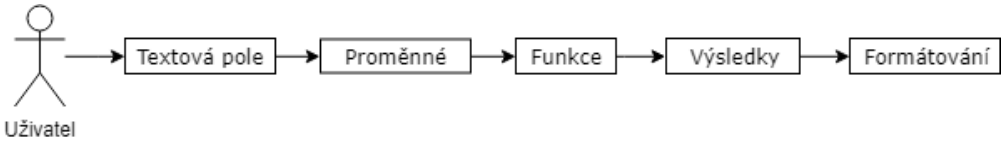
4.1 Požadavky

Obecným požadavkem byl návrh a vývoj doplňku pro aplikaci Microsoft Excel. Jak bylo zmíněno v teoretické části, je vhodné nejdřív provést kratší analýzu, co děláme a jak to budeme dělat. Pokusíme se provést i několik abstraktních návrhů pomůcky, a to později převedeme do kódu. Předáním řešení konečnému uživateli ale neznamená konec práce. Aby se zajistila správná funkčnost a uživatel byl spokojen s nástrojem, je vhodné ho obeznámit s možnostmi spuštění, grafickým rozhraním, ovládáním nebo například řešením potíží. Pro zvědavé uživatelé budou demonstrované i ukázky kódu VBA z vývojářského prostředí a k nim bude přiložen krátký komentář.

4.1.1 Vstupní analýza

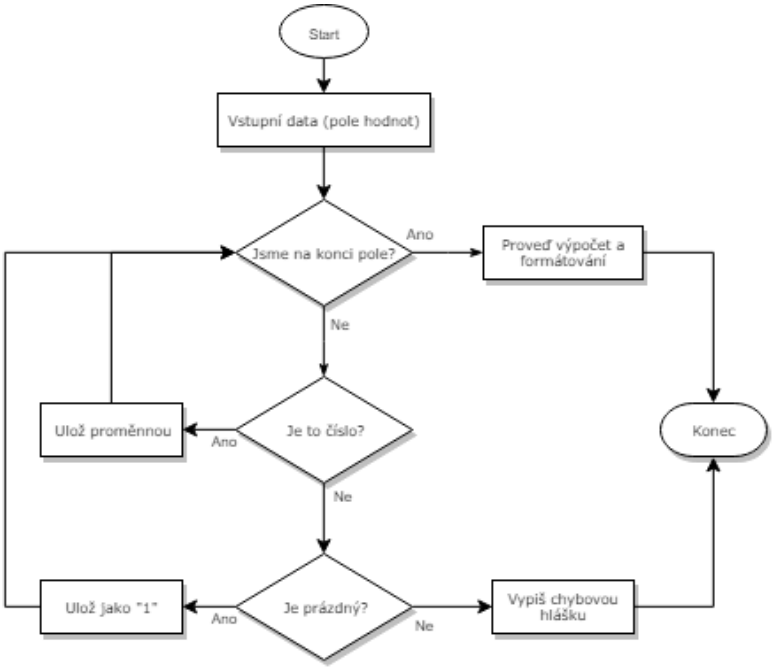
Nejprve je důležité si říci, kdo bude pomůcku používat. Odpověď je: primárně studenti, například pro kontrolu výsledků počtů v ruce. Dále možná učitelé, nadšenci do Excelu nebo nějaký začínající podnikatel, který bude potřebovat orientační čísla pro zásobování svého skladu. Nemůžeme ale předpokládat, že umí většinu funkcí Excelu, metody řízení zásob a už vůbec ne VBA. Proto je zapotřebí vytvořit uživatelské rozhraní UI, které bude pro koncového uživatele intuitivní a srozumitelné. Jako pomoc nám poslouží integrované rozhraní pro tvorbu modální oken, které v editoru jsou v sekci „Forms“. Jedna věc je vytvořit tlačítkové ovládání, věc druhá je převést funkce do VBA a propojit je s tlačítky a textovými poli takovým způsobem, aby to generovalo správné výsledky. Jako pomoc v realizaci nám poslouží několik diagramů.

Pro základní inicializaci se zvolil postup, kde otevřený sešit obsahuje jediný list s uvítáním a pokynem ke spuštění. Tento přístup se zvolil kvůli tomu, aby uživatel nemusel zasahovat do nastavení aplikace Excel a přidávat někam na pracovní panel ikonku pro volání hlavního menu. Ještě se mohl použít přístup, kde by se menu samo otevřelo při startu, ale to by limitovalo uživatele na jediný výpočet a pro další výpočty by se musel soubor otevírat znovu. Jako vstup z uživatelské strany jsou v modálních oknech (volaných z hlavního menu) vytvořená pole pro zadávání textu. Textem je zde myšlen vstup z klávesnice a jelikož se jedná o matematické operace, očekává se, že uživatel bude zadávat čísla.



Obrázek 6: základní postup používání (zdroj: vlastní tvorba)

Uživatel tedy napíše něco do polí a potvrdí tlačítkem Ok. Hodnoty z polí se uloží do proměnných, které jsou dále zpracovány funkcemi, vygenerují se výsledky a ty se ještě naformátují. Pokud ovšem dojde k překlepu nebo někdo bude úmyslně zadávat nevhodné znaky, aplikace výpočet přeruší a vyhodí chybovou hlášku. S lidskou hloupostí se musí počítat vždy, protože pokud některé sofistikované systémy nemají proti tomu ošetření, může to vést k velkým nepříjemnostem jako je ztráta dat, poškození zařízení apod. Pro tento účel tedy vytvoříme nějaký postup pro kontrolu dat. V kódu je při každém potvrzení zadání hodnot provedená kontrola vstupů pomocí funkce *Numeric_Test()*. Ta projde rozsah proměnných a jednu po druhé zkontroluje, zda je numerického charakteru. Pokud není, zkontroluje, zda vůbec do pole uživatel něco zadal. Pokud tedy tam bude nějaký znak, který není numerický a zároveň pole není prázdné, musí se jednat o jiný druh znaků a proto aplikace zastaví výpočet a upozorní uživatele hláškou „Alespoň jeden vstup není numerický.“ V jiném případě se doplní do prázdného pole jednička, a to z důvodu, že se spousta proměnných vyskytuje nejen v čitateli, ale i v jmenovateli a dělení nulou často vyvolá zastavení počítání. Algoritmus funkce by se dal znázornit takto:



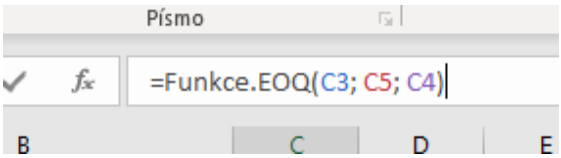
Obrázek 7: kontrola vstupů uživatele (zdroj: vlastní tvorba)

V kódu je toto ošetření realizováno v této podobě:

```
Sub Numeric_Test()

    Dim Ncell As Range
    Dim IsNotNumber As Boolean

    For Each Ncell In Range("C3:C5")
        If Not IsNumeric(Ncell) Then
            IsNotNumber = True
        End If
        If IsEmpty(Ncell) Then
            Ncell = 1
            MsgBox ("Prázdný vstup, doplněná hodnota 1.")
        End If
    Next Ncell

    For Each Ncell In Range("C6:C7")
        If Not IsNumeric(Ncell) Then
            IsNotNumber = True
        End If
        If IsEmpty(Ncell) Then
            Ncell = 0
        End If
    Next Ncell

    Select Case IsNotNumber
    Case True
        MsgBox ("Alespoň jeden vstup není numerický.")
        End
    End Select

End Sub
```

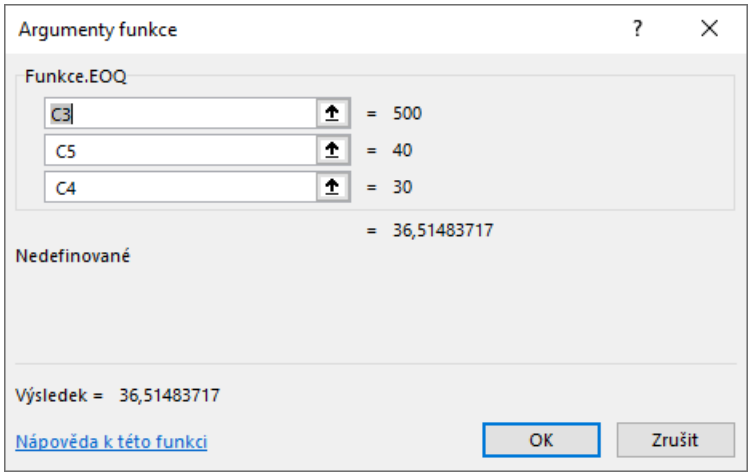
Pro pohodlí uživatele nebudeme ho nutit po každé změně jedné proměnné provádět znovu zadávání hodnot a provádět výpočet. Zadané hodnoty se uloží do autorem definovaných buněk a do předchozího sloupce daných řádků se uloží název proměnné. Samotné výpočty, které jsou převedeny do kódu se pak počítají již z hodnot uložených v buňkách listu sešitu Excelu. To nám umožňuje využít této interaktivity a změnit proměnnou za chodu a dostat ihned přepočítaný výsledek.

Jednotlivé funkce pro výpočet jsou definovány jako uživatelské funkce Excelu. To znamená, že pokud by nebyly autorem skryté, lze je volat z řádku pro zadávání funkcí a zapisovat do nich příslušné argumenty funkce. Pokud bychom to měli ukázat na příkladu, vypadalo by to takto:



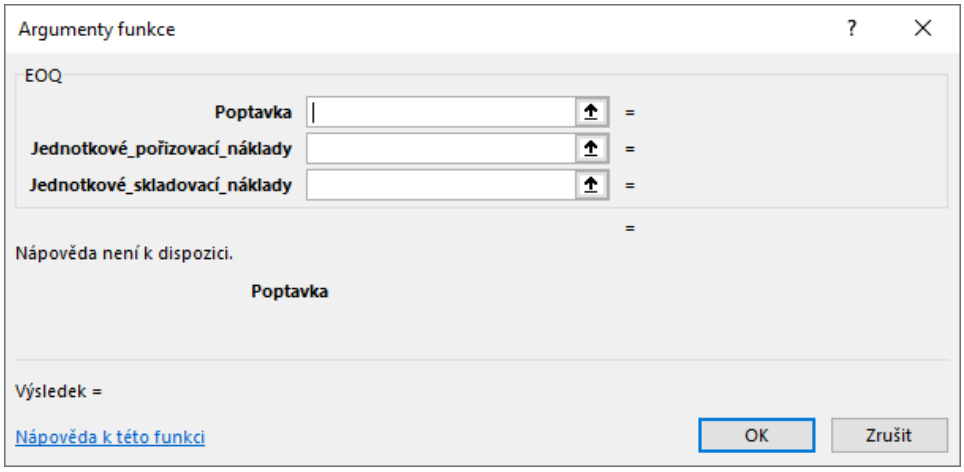
Obrázek 8: podoba uživatelské funkce (zdroj: vlastní tvorba)

Tím, že je funkce uložená v modulu pojmenovaném „Funkce“ a VBA využívá tečkovou konvenci, můžeme z něj příslušnou funkci zavolat a vložit do ní adresy buněk nebo pevné hodnoty. Název modulu před funkcí není povinný, ale pokud by se vložil další modul se stejnou funkcí, docházelo by k problémům. Horší to však je s vytvářením popisů a nápovědy pro modální okno. Proto po zavolání funkce vypadá pouze takto:



Obrázek 9: podoba uživatelské funkce volané z modulu (zdroj: vlastní tvorba)

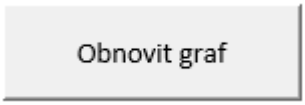
Když ale zavoláme tu samou funkci bez modulu, ve kterém má být hledaná (pouze v podobě *=EOQ(argument1,...)*), dostaneme tuto nabídku:



Obrázek 10: podoba uživatelské funkce s názvy proměnných (zdroj: vlastní tvorba)

Někdo by mohl vytknout, že to není šťastné řešení a porušuje základní doporučení vývoje SW. Na druhou stranu výstup práce primárně slouží ke studijním účelům, a proto autor si dovolil proměnné nadefinovat česky, a to rovnou do těla funkce. Navíc dle Krále (2010) by vlastní funkce nikdy nedosahovali stejné rychlosti, jako již nadefinované v základu. Dále je nutno vytknout, že i zde by se dala provést modifikace, pokud někdo bude z této práce vycházet a pokračovat s její rozšiřováním.

Některé metody a jejich výpočty se dají znázornit i graficky. Pro tento účel je v nich zakomponovaná funkce pro vložení matice čísel, ze které se vykreslí graf. V kódu se jedná o několik funkcí (v závislosti na metodě), které mají pojmenování *fillCells*()* (kde * je nic nebo celé číslo) a ty provádějí přepočítání dané metody pro násobky optimální hodnoty. Násobky jsou generovány od 20 %, přes aktuální hodnotu, až po 180%. Táto škála pak slouží jako vstup pro výpočet ostatních proměnných. Tato matice čísel se následně označí a pomocí příkazu pro aplikaci Excel (*ActiveSheet.Shapes.AddChart2(332, xlLineMarkers).Select*) se vloží na aktivní list graf včetně legendy. Graf je umístěn na přesné místo listu a zakrývá tak matici čísel, aby to nemátlo uživatele. Kvůli riziku zacyklení se bohužel nedalo zakomponovat do počítání již definované funkce, proto jsou výpočty prováděné natvrdo. Pokud tedy změníme nějakou proměnnou, musí se matice přepočítat. Pro tento účel se na list vkládat aktivní prvek v podobě tlačítka:



Obrázek 11: tlačítko pro přepočet matice vstupů grafu (zdroj: vlastní tvorba)

Tímto tlačítkem se v podstatě znovu spustí funkce *fillCells*()*. Na závěr se mimo vložení již zmíněných komentářů k buňkám se vstupní hodnoty zvýrazní žlutou barvou a výstupní zelenou.

4.1.2 Formát a spuštění nástroje

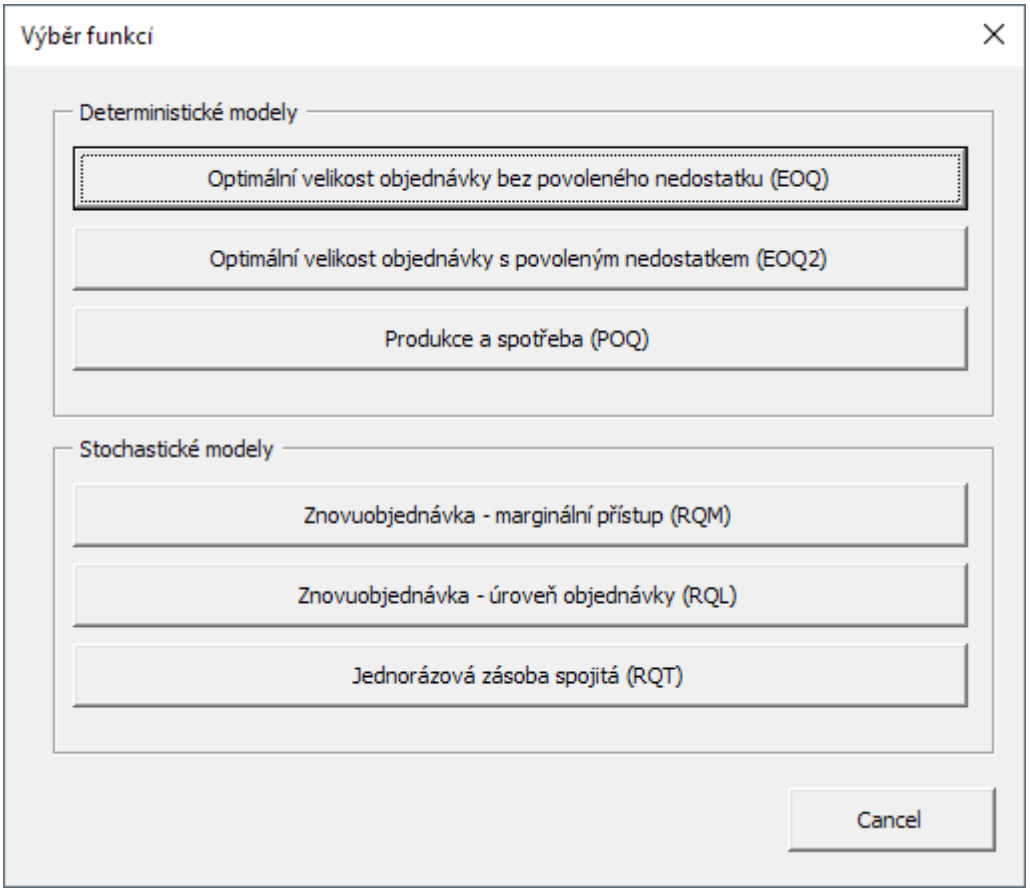
Výsledkem této práce je soubor typu .xlsm, který se definuje jako „Sešit Excelu s podporou maker“. Měl by být podporován ve verzích aplikace 2010, 2013, 2016 a Office 365. Testován byl na posledních třech jmenovaných. Jak již bylo specifikováno v teoretické části, doplněk představuje samostatnou část prostředí aplikace a od spustitelného souboru typu .xls (resp. .xlsx) se liší tím, že se napojuje z adresáře a zůstává aktivní, dokud není napojení přerušeno uživatelem nebo se dané rozšíření nepřemístí.

Výchozím umístěním pro jmenované prvky je skrytý uživatelský adresář obsahující nastavení aplikací (například pro prostředí Windows 10 to je C:\Users*jméno_uživatele*\AppData\Roaming\Microsoft\AddIns). Když ale vezmeme v úvahu fakt, že ve školním nebo firemním prostředí se často používá řešení ve verzi Enterprise, musel by se doplněk implementovat plošně pro celou síť uživatelů. To by mohlo porušovat bezpečnostní politiky daného subjektu nebo přinejmenším vyžadovalo dodatečnou analýzu a jednání s aplikačním gestorem, což nebylo předmětem diplomové práce.

Druhý případ je stahovat doplněk z různých zdrojů, například z portálu pro podporu výuky, moodle.czu.cz. Řešení jednoduché leč nepraktické, jelikož vyžaduje po uživateli manuální napojení doplňku v nastavení aplikace. Po odhlášení se uživatelské prostředí resetuje a doplněk se musí znovu stahovat a napojovat. Proto autor práce přistoupil na zprovoznění nástroje v již zmíněném formátu běžného sešitu aplikace Excel s podporou maker. Soubor se otevírá jako běžný sešit Excelu s tou výjimkou, že po spuštění musí uživatel povolit makra. Většinou se to v aplikaci projeví žlutou lištou pod hlavní nabídkou Excelu, která vyzývá uživatele k povolení obsahu.

4.1.3 Grafické prostředí a ovládání

Po spuštění aplikace se uživateli zobrazí sešit Excelu, který obsahuje jediný list pojmenovaný „Menu“. Na něm je vyzván ke kliknutí na aktivní prvek (dále jen tlačítko) pojmenovaný „Spustit“. Nahrazuje to ikonu v horní nabídce nástrojů, která by se objevila v případě realizace doplňku. Po kliknuti se uživateli zobrazí následující nabídka:



Obrázek 12: hlavní nabídka modelů (zdroj: vlastní tvorba)

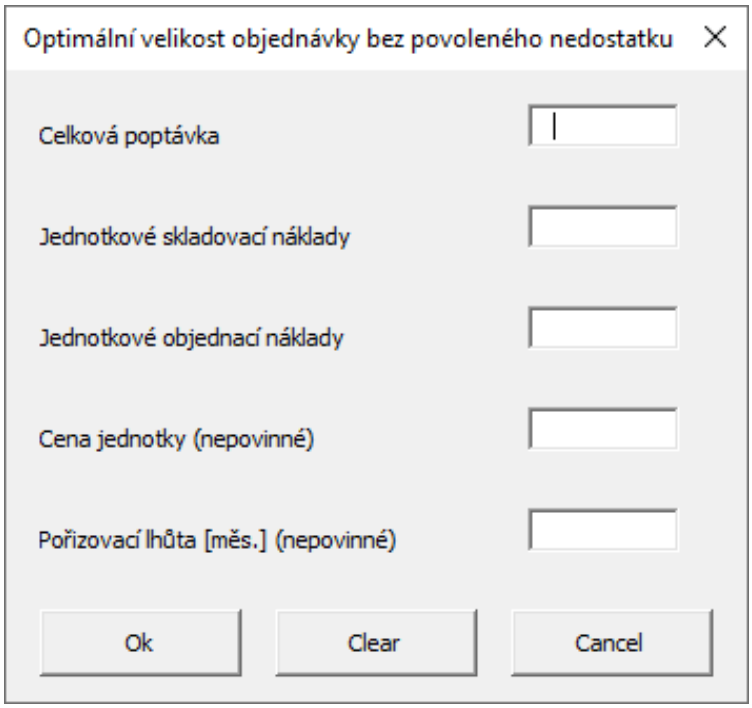
Nabídka je vizuálně rozdělená na dvě sekce, první se věnuje modelům s deterministickou poptávkou, druhá se stochastickou poptávkou. V nástrojích pro vývojaře se pomocí nastavení pořadí prvků (Tab Order) jednotlivá tlačitka seřadila za sebou, což umožňuje přepínat mezi nimi i pomocí šipek na klávesnici uživatele. V tomto případě nahorů a dolů nebo pomocí klavesy Tab. Pokud se uživatel chce vrátit, slouží k tomu standardní ikonka křížku v rohu aktivního okna nebo tlačítko „Cancel“. Po zvolení daného modelu má uživatel možnost zadát hodnoty pro výpočet řešení modelu.

4.2 Popis jednotlivých modelů a jejich struktury

Nástroj obsahuje celkem 6 modelů, jejichž výstupem není jen optimální objednávané množství, ale například i celkové náklady. Po kliknutí na vybraný model se u každého zobrazí vlastní modální okno, kam uživatel zadává hodnoty. Všechna okna obsahují vícero textových polí s popisky a tři tlačítka („OK“, „Clear“ a „Cancel“), pomocí který může uživatel potvrdit své zadání, vyčistit formulář nebo se vrátit na předchozí nabídku.

4.2.1 Optimální velikost objednávky bez povoleného nedostatku (EOQ)

Jak již z názvu vyplývá, model počítá optimální objednávané množství základního modelu. Na výběr máme velikost celkové poptávky, kde bereme v potaz celkovou roční poptávku, dále jednotkové skladovací a objednací náklady. Obsahuje to i pole pro zadání ceny jednotky, která sice v modelu nehraje zásadní roli, ale slouží pro výpočet celkových nákladů za „manipulaci“ s jednotkami včetně jejich pořizovací ceny.



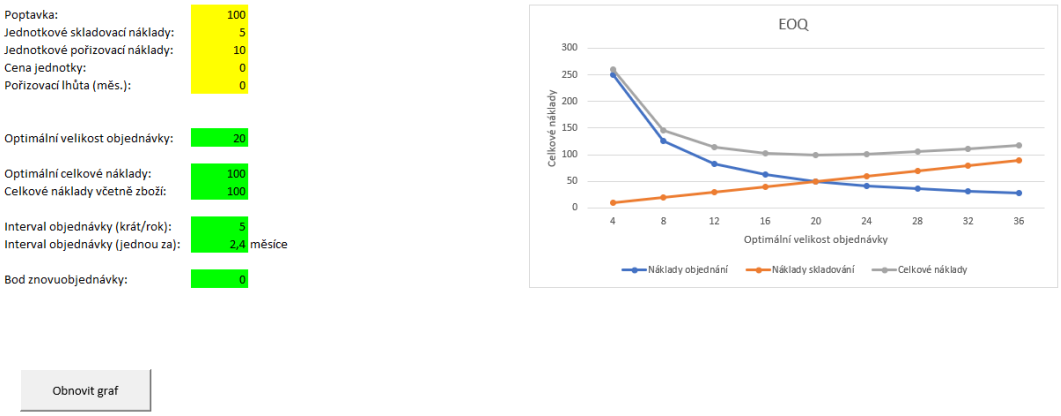
Obrázek 13: nabídka modelu bez povoleného nedostatku (zdroj: vlastní tvorba)

Po zadání číselných hodnot dojde k provedení výpočtu. Ten probíhá dle dvou uživatelem definovaných funkcích, které zajišťují automatický přepočet při změně hodnoty, tak jak to bylo zmíněno v teoretické části. V jazyce VBA se funkce definují tímto kódem:

```
'--- MODEL EOQ bez povoleného nedostatku ---
Function EOQ(Poptavka, Jednotkové_pořizovací_náklady, Jednotkové_skladovací_náklady)
EOQ = Application.WorksheetFunction.Power(((2 * Poptavka * Jednotkové_pořizovací_náklady) / Jednotkové_skladovací_náklady), 1 / 2)
End Function

Function TC_EOQ(Poptavka, Jednotkové_pořizovací_náklady, Jednotkové_skladovací_náklady)
TC_EOQ = Application.WorksheetFunction.Power(2 * Poptavka * Jednotkové_pořizovací_náklady * Jednotkové_skladovací_náklady, 1 / 2)
End Function
```

Jak funkce EOQ tak i TC_EOQ se dají nalézt i mezi běžnými funkcemi Excelu a jsou fyzicky vložené do daných buněk. Jediný zásadní rozdíl je v tom, že jsou zařazené do sekce uživatelem definovaných funkcí. Pro jejich volání je zapotřebí zadat předvolbu podle názvu modulu ve kterém se nacházejí, což by ve výsledku vypadlo jako Funkce.EOQ(argument1; argument2, argument3).

Daný výstup po potvrzení zadání v daném modelu vypadá například takto:



Obrázek 14: výstup modelu bez povoleného nedostatku (zdroj: vlastní tvorba)

Ve všech modelech ve žlutě zvýrazněných buňkách se vyskytují zadané hodnoty, v zeleně zvýrazněných jsou pak výsledky. Některé modely včetně tohoto obsahují i vložený graf pro lepší znázornění principu modelu. V tomto případě se jedná o minimalizaci nákladů za skladování a pořízení, které se v tomto modelu mají rovnat. Jelikož vstupní data grafu se generují vlastní funkcí pomocí cyklů, vkládá se do listu tlačítko, které graf v případě potřeby obnoví. Je to vhodné pro případy, kdy chceme ladit výsledek modelu manuální úpravou hodnot ve žlutě vyznačených buňkách.

Funkce pro generování grafu mají tuto podobu:

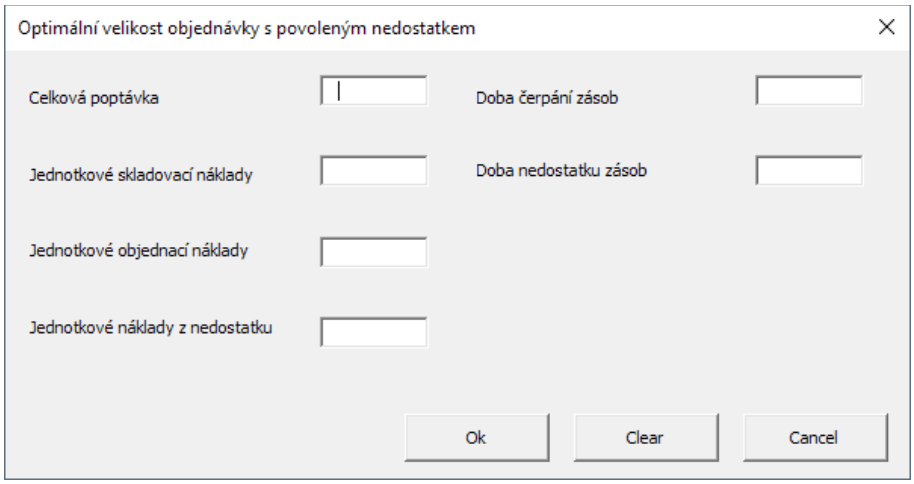
```
Private Sub fillCells()
    Application.Volatile
    Dim i As Integer
    For i = 1 To 9
    'Cells(i + 4, 9) = i
    Cells(i + 4, 10) = Cells(10, 3) * i * 0.2
    Cells(i + 4, 11) = Cells(3, 3) / Cells(i + 4, 10) * Cells(5, 3)
    Cells(i + 4, 12) = Cells(i + 4, 10) / 2 * Cells(4, 3)
    Cells(i + 4, 13) = Cells(i + 4, 11) + Cells(i + 4, 12)
    Cells(i + 4, 14) = Cells(i + 4, 11) + Cells(i + 4, 12) + Cells(3, 3) * Cells(6, 3)
    Next i
End Sub
```

a

```
Range("J9:N9").Interior.ColorIndex = 4
Range("J4:N14").Select
Selection.NumberFormat = "0"
ActiveSheet.Shapes.AddChart2(332, xlLineMarkers).Select
With ActiveChart.Parent
  .Width = 450
  .Height = 240
  .Top = Range("I3").Top
  .Left = Range("I3").Left
End With
ActiveChart.SetSourceData Source:=Range("$K$4:$M$13")
ActiveChart.ChartTitle.Text = "EOQ"
ActiveChart.FullSeriesCollection(1).XValues = "='" & ActiveSheet.Name & "'!$J$5:$J$13"
```

4.2.2 Optimální velikost objednávky s povoleným nedostatkem (EOQ2)

Druhý model počítá s povoleným nedostatkem zásoby. Chybějící zásoba se projeví hlavně v celkových nákladech, které jsou zároveň silně ovlivněny i intervaly dostatku a nedostatku zásoby.



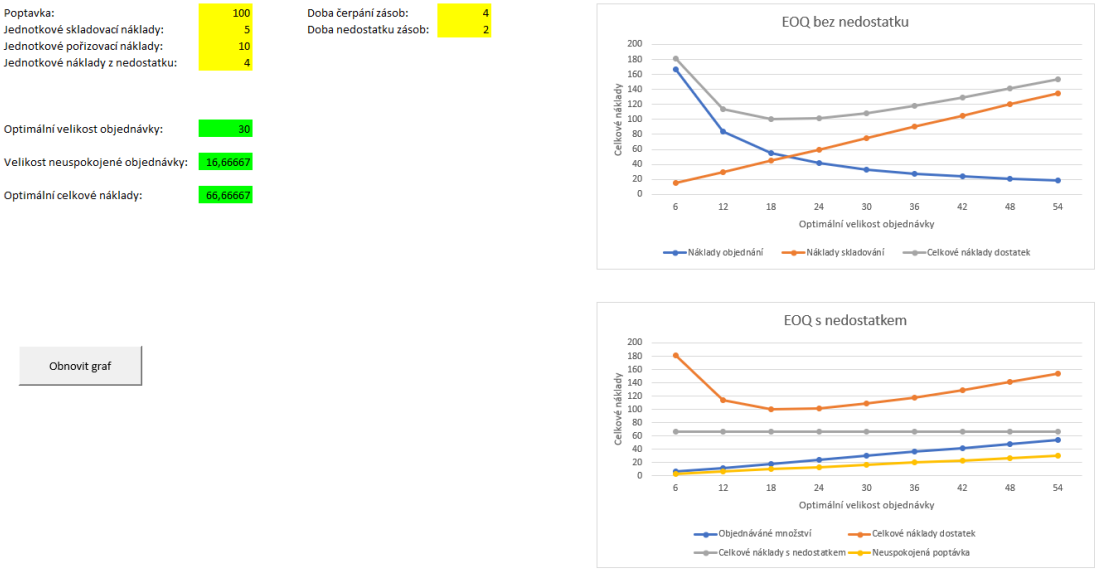
Obrázek 15: nabídka modelu s povoleným nedostatkem (zdroj: vlastní tvorba)

Výpočty tohoto modelu jsou definovany funkcí pro výpočet optimálního množství shodnou s prvním modelem, dále funkci na výpočet neuspokojené poptávky a funkci celkových nákladů:

```
'--- MODEL EOQ s povoleným nedostatkem ---
Function EOQ2(Poptavka, Jednotkové_pořizovací_náklady, Jednotkové_skladovací_náklady,
Jednotkové_náklady_neuspokojení)
EOQ2 = Application.WorksheetFunction.Power(((2 * Poptavka * Jednotkové_pořizovací_náklady) /
Jednotkové_skladovací_náklady), 1 / 2) *
Application.WorksheetFunction.Power(((Jednotkové_skladovací_náklady +
Jednotkové_náklady_neuspokojení) / Jednotkové_náklady_neuspokojení), 1 / 2)
End Function
```

```
Function S_EOQ2(Q, Jednotkové_skladovací_náklady, Jednotkové_náklady_neuspokojení) 'množství neuspokojené poptávky
S_EOQ2 = Q * Jednotkové_skladovací_náklady / (Jednotkové_skladovací_náklady + Jednotkové_náklady_neuspokojení)
End Function

Function TC_EOQ2(Poptavka, Q, s, t1, t2, Jednotkové_pořizovací_náklady, Jednotkové_skladovací_náklady, Jednotkové_náklady_neuspokojení)
TC_EOQ2 = (Poptavka / Q) * (Jednotkové_pořizovací_náklady + (Jednotkové_skladovací_náklady * (Q - s) / 2) * t1 + (Jednotkové_náklady_neuspokojení * s / 2 * t2))
End Function
```

Stejně jako u ostatních modelů i tyto funkce se dají nalézt v nabídce funkcí aplikace Excel. Výsledkem tohoto modelu je znovu několik vyplněných buněk a graf.

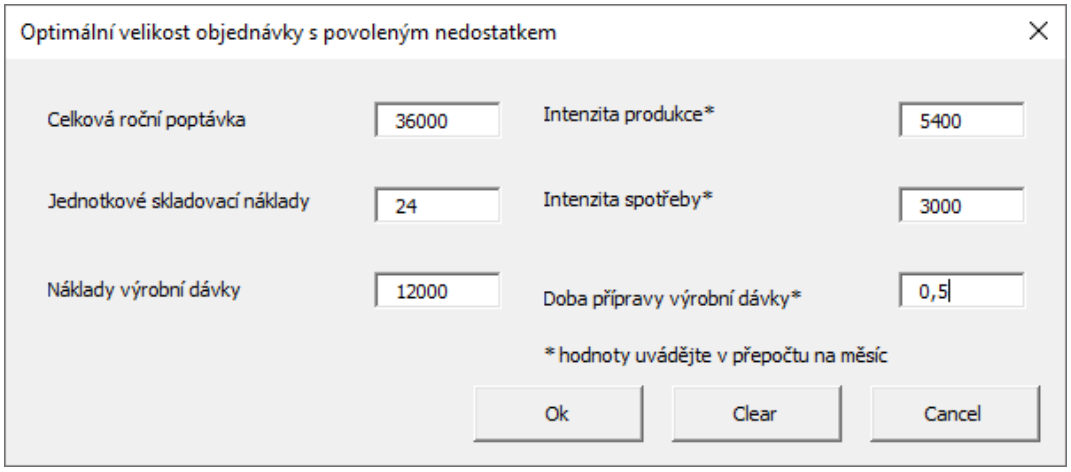


Obrázek 16: výstup modelu s povoleným nedostatkem (zdroj: vlastní tvorba)

Výsledky modelu znovu nalezneme v zeleně vyzněčených buňkách listu. Graf vychází z předchozího modelu pro znázornění nákladů. Pokud bychom porovnávali celkové náklady modelu EOQ s dostatkem a nedostatkem zásoby, zjistíme že náklady druhého modelu jsou nesrovnatelně vyšší než u prvního modelu. To je způsobeno nákladem z nedostatku jednotky a dále i dobami čerpání a nedostatku zásob. V případnem grafu by se to projevilo například dvěma horizontálními křivkami daleko vzdálenými od sebe, proto nápad na zmíněný případný graf byl během vývoje nástroje zavržen.

4.2.3 Model produkce a spotřeby

Třetí a poslední model s deterministickou poptávkou pracuje s poměrem intenzity výroby a spotřeby jednotek. Dalšími proměnnými, které mají zásadní vliv na výsledek celkových nákladů jsou náklady výrobní dávky a délky cyklů spotřeby a výroby.



Obrázek 17: nabídka modelu produkce a spotřeby (zdroj: vlastní tvorba)

Pro tento model jsou v modulu definovany následující funkce:

```
'--- MODEL produkce a spotřeby ---
Function POQ(Poptavka, Intenzita_produkce, Intenzita_spotřeby, Náklady_výrobní_dávky, Jednotkové_skladovací_náklady)
POQ = Application.WorksheetFunction.Power(((2 * Poptavka * Náklady_výrobní_dávky) / Jednotkové_skladovací_náklady), 1 / 2) * Application.WorksheetFunction.Power((Intenzita_produkce / (Intenzita_produkce - Intenzita_spotřeby)), 1 / 2)
End Function

Function TC_POQ(Poptavka, Intenzita_produkce, Intenzita_spotřeby, Náklady_výrobní_dávky, Jednotkové_skladovací_náklady)
TC_POQ = Application.WorksheetFunction.Power((2 * Poptavka * Náklady_výrobní_dávky * Jednotkové_skladovací_náklady), 1 / 2) * Application.WorksheetFunction.Power(((Intenzita_produkce - Intenzita_spotřeby) / Intenzita_produkce), 1 / 2)
End Function
```

Celková roční poptavka:	36000	Intenzita měsíční produkce:	5400
Jednotkové skladovací náklady:	24	Intenzita měsíční spotřeby:	3000
Náklady výrobní dávky:	12000	Doba přípravy výrobní dávky:	0,5

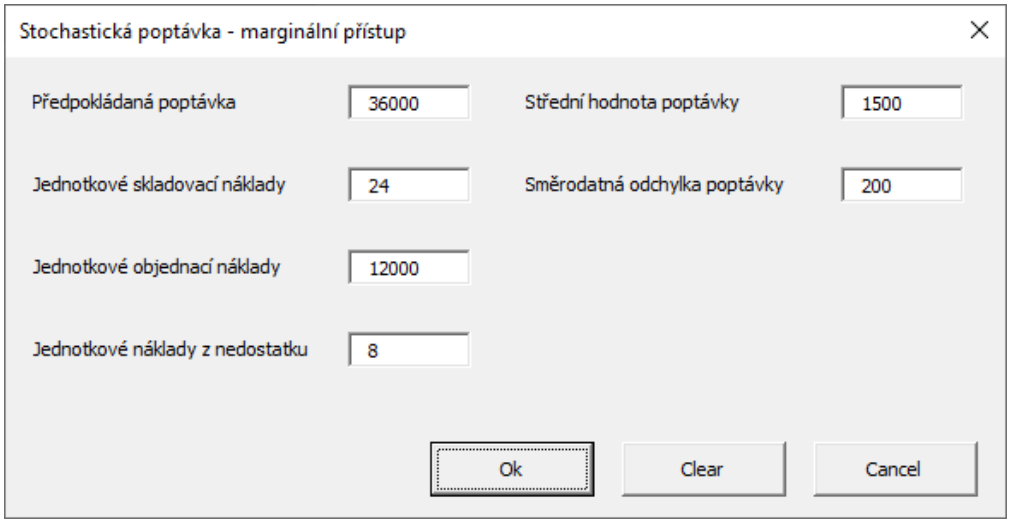
Optimální objem výrobní dávky:	9000 kusů	Výrobní cyklus:	1,666667 měsíců
Optimální délka intervalu mezi dodávkami:	3 měsíců	Spotřební cyklus:	1,333333 měsíců
Celkové náklady výroby:	96000		
Maximální zásoba na skladě:	4000 kusů		
Úroveň zásoby pro přípravu nové dávky:	1500 kusů		

Obrázek 18: výstup modelu produkce a spotřeby (zdroj: vlastní tvorba)

Výstupem modelu není jen optimální objem výrobní dávky a celkové náklady, ale i délky výrobních cyklů, maximální zásoba skladu nebo úroveň zásob pro přípravu nové dávky. Pokud by někdo vycházel z této práce, má možnost naprogramovat k modelu grafický výstup. To by zahrnovalo funkce, které z výsledných hodnot generovali vstupní data grafu. Pro daný model by se jednalo o jednorázovou produkci zásoby a její postupnou spotřebu. Aplikace Excel v současné době nenabízí typ grafu, který by ideálně odpovídal potřebě uživatele pro demonstraci principu modelu.

4.2.4 Stochastická poptávka s marginálním přístupem

Nyní se práce posouvá do sekce modelů se stochastickou poptávkou. Prvním zástupcem modelů je s marginálním přístupem.

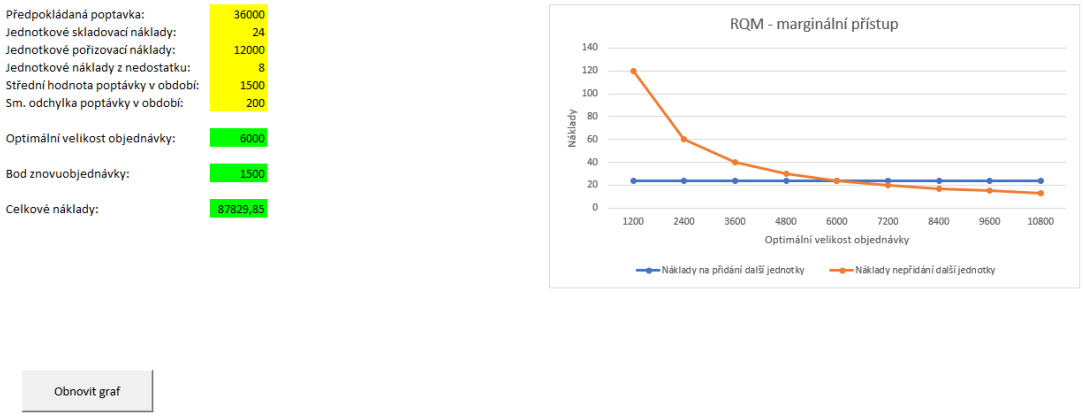


Obrázek 19: nabídka modelu s marginálním přístupem (zdroj: vlastní tvorba)

Po zadání příslušných číselných hodnot dostaneme tři výsledky a graf. Výslekdy jsou generovány těmito funkcemi, které jsou rovněž dostupné i mimo menu.

```
'--- MODEL se stochastickou poptávkou - marginální přístup (RQM)---
Function RQM(Poptavka_odhad, Jednotkové_pořizovací_náklady, Jednotkové_skladovací_náklady)
RQM = Application.WorksheetFunction.Power(((2 * Poptavka_odhad * Jednotkové_pořizovací_náklady) / Jednotkové_skladovací_náklady), 1 / 2)
End Function


Function RQM_Reorderpoint(Jednotkové_skladovací_náklady, Jednotkové_náklady_neuspokojení, Optimální_množství, Poptavka_odhad, Sigma_M, Střední_hodnota_poptávky)
FR = 1 - ((Jednotkové_skladovací_náklady * Optimální_množství) / (Jednotkové_náklady_neuspokojení * Poptavka_odhad))
Z = Application.WorksheetFunction.Norm_S_Inv(FR)
```

```
w = Z * Sigma_M
RQM_Reorderpoint = Střední_hodnota_poptávky + w
End Function

Function TC_RQM(Jednotkové_pořizovací_náklady, Jednotkové_náklady_neuspokojení, Sigma_M, Jednotkové_skladovací_náklady, Optimální_množství, Poptavka_odhad, Střední_hodnota_poptávky, Bod_znovuobjednávky)
FR = 1 - ((Jednotkové_skladovací_náklady * Optimální_množství) / (Jednotkové_náklady_neuspokojení * Poptavka_odhad))
Z = Application.WorksheetFunction.Norm_S_Inv(FR)
NZ = Application.WorksheetFunction.Norm_S_Dist(Z, False)
TC_RQM = (Jednotkové_pořizovací_náklady + (Jednotkové_náklady_neuspokojení * Sigma_M * NZ * (Poptavka_odhad / Optimální_množství))) + (((Optimální_množství / 2) + (Bod_znovuobjednávky - Střední_hodnota_poptávky)) * Jednotkové_skladovací_náklady)
End Function
```

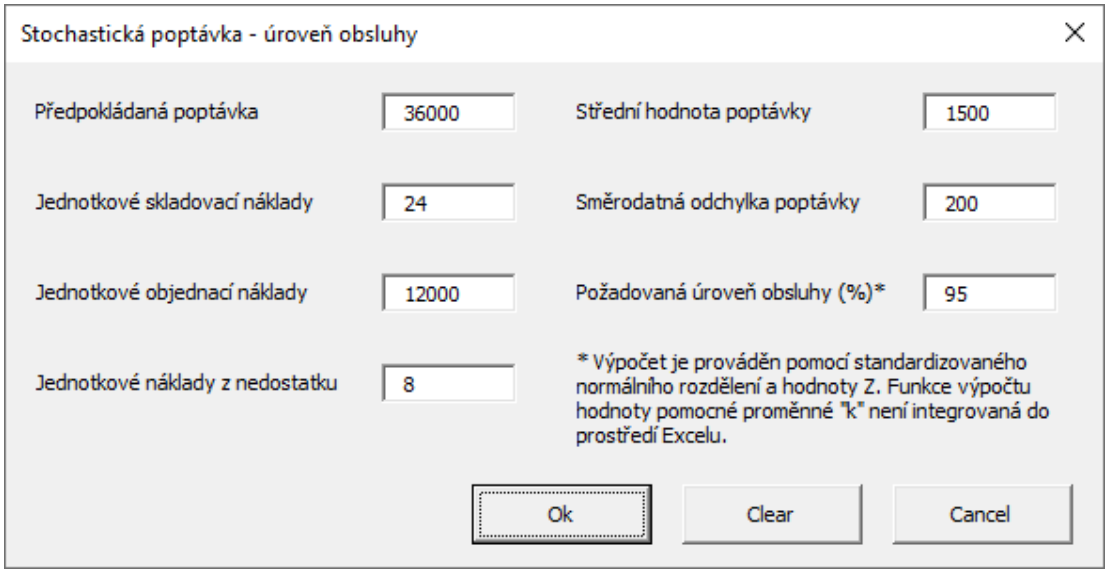


Obrázek 20: výstup modelu s marginálním přístupem (zdroj: vlastní tvorba)

Jak je možné vidět na tomto výstupu, tak kromě tří výsledků modelu mám k dispozici znovu graf, který demonstruje princip modelu, kde se vyrovnávají náklady na přidání a nepřidání další jednotky. Přiložený graf si musíme v případě změny vstupních hodnot manuálně obnovit.

4.2.5 Stochastická poptávka s určením úrovně objednávky

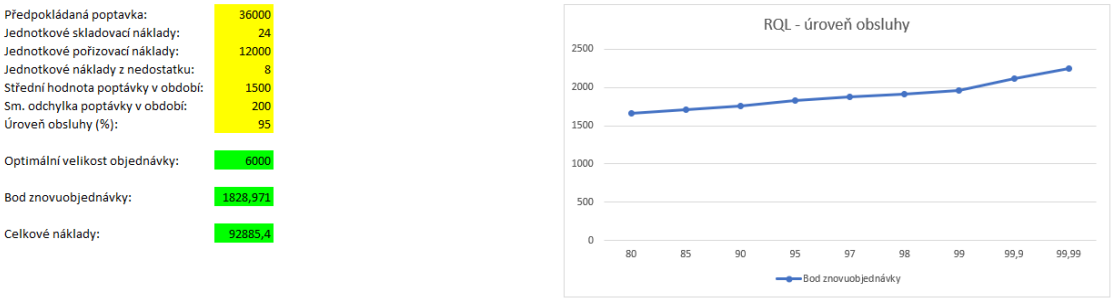
Na rozdíl od předchozího modelu v tomto si uživatel určuje o jednu vstupní proměnnou navíc, a to úroveň obsluhy. Hodnota se zadává v číslech z intervalu 1 až 100. Pokud se zadá jiná hodnota, uživatel bude na to upozorněn vyskakovacím oknem.



Obrázek 21: nabídka modelu s určením úrovně objednávky (zdroj: vlastní tvorba)

Funkce jsou definovány pomocí těchto zdrojových kódů:

```
'--- MODEL se stochastickou poptávkou - úroveň objednávky(RQL)---
Function RQL(Poptavka_odhad, Jednotkové_pořizovací_náklady, Jednotkové_skladovací_náklady)
RQL = Application.WorksheetFunction.Power(((2 * Poptavka_odhad * Jednotkové_pořizovací_náklady) / Jednotkové_skladovací_náklady), 1 / 2)
End Function

Function RQL_Reorderpoint(Jednotkové_skladovací_náklady, Jednotkové_náklady_neuspokojení, Optimální_množství, Poptavka_odhad, Sigma_M, Střední_hodnota_poptávky, Úroveň_objednávky)
Z = Application.WorksheetFunction.Norm_S_Inv(Úroveň_objednávky / 100)
w = Z * Sigma_M
RQL_Reorderpoint = Střední_hodnota_poptávky + w
End Function

Function TC_RQL(Jednotkové_pořizovací_náklady, Jednotkové_náklady_neuspokojení, Sigma_M, Jednotkové_skladovací_náklady, Optimální_množství, Poptavka_odhad, Střední_hodnota_poptávky, Bod_znovuobjednávky, Úroveň_objednávky)
Z = Application.WorksheetFunction.Norm_S_Inv(Úroveň_objednávky / 100)
NZ = Application.WorksheetFunction.Norm_S_Dist(Z, False)
TC_RQL = (Jednotkové_pořizovací_náklady + (Jednotkové_náklady_neuspokojení * Sigma_M * NZ * (Poptavka_odhad / Optimální_množství))) + (((Optimální_množství / 2) + (Bod_znovuobjednávky - Střední_hodnota_poptávky)) * Jednotkové_skladovací_náklady)
End Function
```

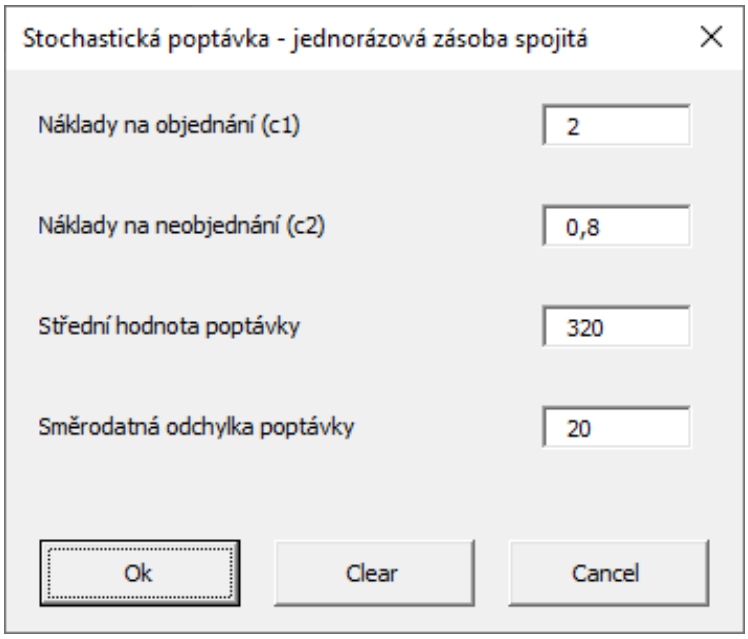


Obrázek 22: výstup modelu s určením úrovně objednávky (zdroj: vlastní tvorba)

Výstup tohoto modelu obsahuje požadované výsledky a graf, který obsahuje pevně stanovené hodnoty úrovně obsluhy. Přepočítává se sám s využitím pevně vložené funkce, proto nevyžaduje dodatečné tlačítko.

4.2.6 Jednorázová spojitá poptávka

Poslední model se zabývá jednorázovou zásobou. Jelikož jeho princip spočívá v poměru hodnoty objednání a neobjednání další jednotky, nevyžaduje tolik proměnných. V potaz se bere normální rozdělení, kde se pomocí integrovaných statistických tabulek přepočítává koeficient, kterým se má vynásobit směrodatná odchylka poptávky a pak přičíst ke střední hodnotě poptávky.



Obrázek 23: nabídka modelu jednorázové zásoby se spojitou poptávkou (zdroj: vlastní tvorba)

Pro tento model je definovaná jediná funkce, která má následující podobu:

```
'--- MODEL se stochastickou poptávkou - jednorázově (RQT)---
Function RQT(c1, c2, Střední_hodnota, Sm_odchylka)
    gama = c2 / (c1 + c2)
    Z = Application.WorksheetFunction.Norm_S_Inv(gama)
    RQT = Střední_hodnota + Z * Sm_odchylka
End Function
```

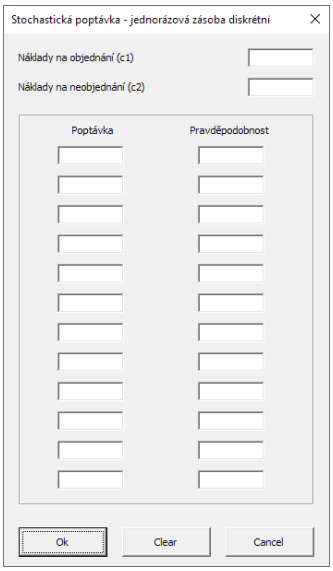
Model vrací pouze jediný výsledek, a to končené množství, které by bylo doporučeno objednat.

Náklady na objednání:	2
Náklady na neobjednání:	0,8
Střední hodnota poptávky:	320
Směrodatná odchylka poptávky:	20
Optimální úroveň objednávky:	308,681

Obrázek 24: výstup modelu s jednorázovou zásobou se spojitou poptávkou (zdroj: vlastní tvorba)

Jednorázová zásoba má i další možné řešení, kde se nepočítá se spojitou poptávkou, ale s diskrétní. Tento model však není uživatelsky přívětivý, protože by vyžadoval zadávat pro každý stupeň poptávky vlastní pravděpodobnost. Rovněž výpočetně by se musela udělat specifická funkce, která by vícenásobným porovnáním hodnot určila, ke které hladině diskrétní poptávky se má přiřadit a vypsat výsledek. To by mohlo být jedním z předmětů případné navazující práce.

Návrh formuláře by mohl vypadát následovně:



Obrázek 25: návrh nabídky modelu jednorázové zásoby s diskrétní poptávkou (zdroj: vlastní tvorba)

5 Výsledky a diskuse

5.1 Programování a jazyk VBA

Když se vrátíme na začátek práce, hlavním cílem bylo vytvořit nástroj, který na základě vstupů uživatele provádí výpočty uživatelem zvolené metody řízení zásob. Jako nástroj se zvolilo vývojářské prostředí v aplikaci Excel od společnosti Microsoft a jazyk VBA. Odůvodněno to bylo tím, že úlohy tohoto typu se většinou počítají pomocí tabulkových procesorů a také tím, že by instalace nebo spuštění externí aplikace mohlo narušovat bezpečnostní politiku IT infrastruktury školy nebo podniku. Rovněž se dbalo i na pohodlí uživatele, aby nemusel nic dodatečně hledat, nastavovat a instalovat. Proto výstupem nakonec není rozšíření aplikace, ale spustitelný soubor, který vyžaduje pouze povolení maker.

V praktické části práce jsou ukázky kódu v jazyce VBA, které znázorňují vybrané funkce a metody přepsané do pseudokódu. Samotný jazyk není nijak složitý a pro pokročilejšího uživatele Excelu by neměl být problém se v něm vyznat. Zdrojový kód není nijak zamčený či skrytý, proto pokud je v aplikaci povolený režim vývojáře, stačí pomocí klávesové zkratky Alt + F11 prokliknout do vývojářského prostředí a kód prozkoumat. Ten není v praktické části z praktických důvodů zdokumentovaný, na druhou stranu obsahuje komentáře, aby byl přehlednější.

Než se pomůcka dostala do stavu, v jakém je teď, byla několikrát předělána. Sice ne úplně od základu, ale měnila se logika vlastních funkcí, přidávali se aktivní prvky, re-designovali modální okna apod. Rovněž pro uživatelské pohodlí optimalizovalo pořadí textových polí a tlačítek, aby například tlačítkem Tab nebo Enter mohlo pohodlně přepínat na další vstupní pole. Přidáním tlačítka Clear se rovněž vyšlo vstříc, a tak pokud někdo nebude spokojený se svým vstupem, může formulář jednoduše vyčistit a začít znovu.

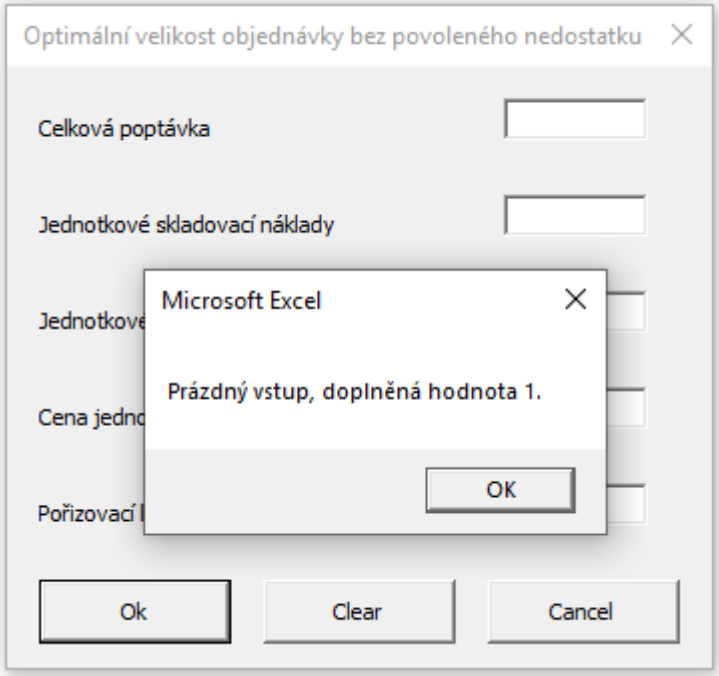
5.2 Pomůcka pro výpočet metod řízení zásob

Samotný výsledek práce autora asi nedosáhne velkého uznání a kvality placených aplikací nebo integrovaných modulů ERP systémů, které už dané, a nejen ty, metody obsahují. Pro studijní účely však přínosný být může. Kalkulátor nakonec pokrývá tři deterministické a tři stochastické metody. Některé z nich obsahují i grafické znázornění pro lepší představu jejich fungování. Pokud uživatel bude postupovat podle pokynů, nemělo by

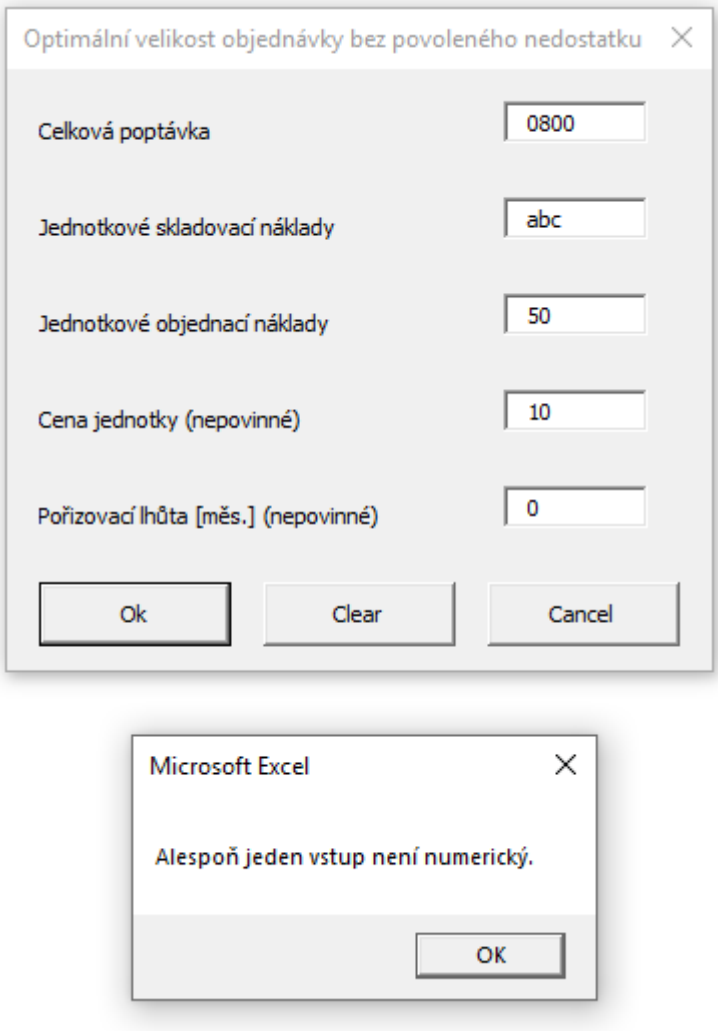
dojít k potížím. Ovšem pokud bude zadávat špatné vstupy, měl by dostávat alespoň upozornění, že dělá něco špatně. Integrované funkce se dají použít i samostatně, mimo pomůcku. Problémem je zapamatovat si jejich zkratky podle definice autora a nutnost rozkliknout modální okno, kde je jasně vidět, na jaké místo funkce vložit jaký argument.

5.3 Testování pomůcky

Pro testování výsledků se použily stejné příklady, jako v teoretické části. Čerpány byly převážně od Jablonského (2007) a celkem reálně znázorňuji možné situace v praxi. Testování je již součástí praktické části, kde jsou zahrnuty v rámci ukázky výstupu, proto se v této kapitolce zaměříme pouze na testování vstupů, které je pro všechny metody relativně stejné, mění se pouze rozsah kontrolovaných buněk. Výjimku tvoří pouze první metoda EOQ, která má dvě nepovinné proměnné a u těch, pokud jsou prázdné, se doplní místo jedničky číslovka nula.



Obrázek 26: pokus s prázdným formulářem (zdroj: vlastní tvorba)



Obrázek 27: pokus s formulářem obsahujícím nečíselné znaky (zdroj: vlastní tvorba)

6 Závěr

V závěru práce se sluší zrekapitulovat, o čem pojednává tato diplomová práce, co bylo její cílem a zda se pomocí vyjmenované metodiky dosáhlo stanoveného cíle. Jako cíl si autor položil tvorbu doplňku do aplikace MS Excel. Cíle se dosáhlo, ale ne podle původní myšlenky. Tak jako každý den narazíme na překážky v cestě, výkonu procesu, prováděni činnosti atd., i zde se vyskytlo několik úskalí, která se musela nějakým způsobem překonat nebo obejít. Nakonec výsledek práce má podobu interaktivního spustitelného souboru aplikace Excel a obsahuje v sobě umělé vytvořené funkce a mechaniky. Ty umožňuji uživateli získat výsledky vybraných metod řízení zásob, jež jsou součástí oboru operačního výzkumu. Autor s prací není zcela spokojen, protože existovalo ještě spousta prostoru, jak pomůcku dále rozšířit a vylepšit. Co však nebylo v dostatečné míře, byla ochota autora, což je celkem škoda. Pokud se na problém podíváme detailněji, lze do pomůcky přidat další metody řešení, funkce se dají napsat čistěji a z programátorského hlediska správněji. Studenti z výměnných pobytů by určitě vedle češtiny uvítali více anglickou či jinou jazykovou verzi, stejně jak tomu bylo například u doplňku LINKOSA. Proto se zde naskytuje možnost na tuto práci dále navázat a práci autora dostat do formy, která bude kompletnější, efektivnější a prospěšnější pro širší skupinu osob.

Když se ale vrátíme k rekapitulaci, vedle metod řízení zásob, které jsou vskutku popsané velmi podrobně v první části práce, čtenář se seznámí trochu i s historii programování/algoritmizace, která je nezbytnou součástí vývoje softwaru. Dále také objeví „skryté“ možnosti Excelu, který je opravdu velmi univerzálním a mocným nástrojem. Pokud by se metodiky a doporučení zmíněné v rešerši používali od začátku, tvorba pomůcky by byla o mnoho méně náročná, než tomu bylo ve skutečnosti. Pro ty, co se nesetkali se systémovou vědou, lze vytknout, že každý systém se dá vylepšit a každý nově navrhovaný systém udělat optimálně efektivní. Proto kdyby se provedla důkladnější analýza a použili správné nástroje a metodiky, mohl být výsledek na lepší úrovni. Nicméně i to minimum použité metodiky bylo užitečnější, než jen „vařit z vody“.

Praktická část, kterou začínal samotný proces tvorby, obeznámila autora s možnostmi vytváření zajímavých, leč užitečných věcí. Nejednalo se sice o pokročilé programování používající třídy (class), vlastní objekty a paměť, ale i tak se muselo dost věcí převést do kódu, nastudovat alespoň základní vlastnosti VBA jazyku, dohledat návody v učebnicích na některé algoritmy a na nich postavené funkce, a celé to skloubit dohromady

tak, aby to poskytovalo uspokojivé výsledky. A jak už bylo zmíněno v úvodu kapitoly, je zde stále dost prostoru pro zlepšování. Dobrou poznámku k tématu měl Albright (2012), kde vysvětluje skutečnost, že programátoři nikdy nejsou dostatečně spokojeni. Pořád si budou pohrávat s kódem, a to nejen, aby fungoval lépe, ale často i proto, aby byl elegantnější a jednodušší na pochopení pro ostatní.

Na úplný závěr – nechť tato práce a její výsledek budou někomu prospěšné, ať už jako inspirace / podklad pro vlastní práci nebo pomůcka na kontrolu výsledků počtů. Byla by škoda, kdyby čas věnovaný na vypracování byl ohodnocen pouze u obhájoby.

7 Seznam použitých zdrojů

[1]. EMMETT, S. *Řízení zásob* : jak minimalizovat náklady a maximalizovat hodnotu. Brno: Computer Press, 2008. ISBN 978-80-251-1828-3.

[2]. HILLIER, F S. -- LIEBERMAN, G J. *Introduction to operations research*. Burr Ridge: McGraw-Hill Higher Education, 2005. ISBN 0-07-321114-1.

[3]. HUŠEK, R. -- MAŇAS, M. *Matematické modely v ekonomii*. Praha: SNTL - Nakladatelství technické literatury, 1989. ISBN 80-03-00098-.

[4]. JABLONSKÝ, J. *Operační výzkum : kvantitativní modely pro ekonomické rozhodování*. Praha: Professional Publishing, 2007. ISBN 978-80-86946-44-3.

[5]. WALKENBACH, J. *Excel 2013 power programming with VBA*. Hoboken: John Wiley & Sons, 2013. ISBN 978-1-118-49039-6.

[6]. WRÓBLEWSKI, P. *Algoritmy : datové struktury a programovací techniky*. Brno: Computer Press, 2004. ISBN 80-251-0343-9.

[7]. HYLMAR, Radek. *Programování pro úplné začátečníky.* Brno: Computer Press, 2009. ISBN 978-80-251-2129-0.

[8]. PROKOP, Jiří. *Algoritmy v jazyku C a C++*. 2., rozš. a aktualiz. vyd. Praha: Grada, 2012. Průvodce (Grada). ISBN 978-80-247-3929-8.

[9]. KRÁL, Martin. Excel VBA: výukový kurz. Brno: Computer Press, 2010. ISBN 978-80-251-2358-4.

[10]. ALBRIGHT, S. Christian. *VBA for Modelers: Developing Decision Support Systems with Microsoft® Office Excel*. South-Western Cengage, 2012. ISBN 978-1-133-19089-9

[11]. URTIS, Tom. *Excel VBA: 24-Hour Trainer*. Wiley, 2011. ISBN 978-0-470-89069-1

[12]. MARTIN, Robert. *Excel Programming with VBA Starter*. Packt Publishing, 2012. ISBN 978-1-84968-844-4

8 Přílohy

8.1 Příloha č.1 – Ukázka zdrojového kódu funkcí

```
'--- Generátor čísel pro sestavení grafu ---
Private Sub fillCells()
    Application.Volatile
    Dim i As Integer
    For i = 1 To 9
    'Cells(i + 4, 9) = i
    Cells(i + 4, 10) = Cells(10, 3) * i * 0.2
    Cells(i + 4, 11) = Cells(3, 3) / Cells(i + 4, 10) * Cells(5, 3)
    Cells(i + 4, 12) = Cells(i + 4, 10) / 2 * Cells(4, 3)
    Cells(i + 4, 13) = Cells(i + 4, 11) + Cells(i + 4, 12)
    Cells(i + 4, 14) = Cells(i + 4, 11) + Cells(i + 4, 12) + Cells(3, 3) * Cells(6, 3)
    Next i
End Sub

Private Sub fillCells4()
    Application.Volatile
    Dim i As Integer
    For i = 1 To 9
    'Cells(i + 4, 9) = i
    Cells(i + 4, 10) = Cells(10, 3) * i * 0.2
    Cells(i + 4, 11) = Cells(3, 3) / Cells(i + 4, 10) * Cells(5, 3)
    Cells(i + 4, 12) = Cells(i + 4, 10) / 2 * Cells(4, 3)
    Cells(i + 4, 13) = Cells(i + 4, 11) + Cells(i + 4, 12)
    Cells(i + 4, 14) = (Application.WorksheetFunction.Power((2 * Cells(3, 3) * Cells(4, 3) * Cells(5, 3)), 1 / 2)) *
Application.WorksheetFunction.Power((Cells(6, 3) / (Cells(6, 3) + Cells(4, 3))), 1 / 2)
    Cells(i + 4, 15) = S_EOQ2(Cells(i + 4, 10), Cells(4, 3), Cells(6, 3))
    Next i
End Sub


Private Sub fillCells2()
    Application.Volatile
    Dim i As Integer
    For i = 1 To 9
    Cells(i + 4, 15) = (Cells(3, 3) / Cells(i + 4, 10)) * (Cells(5, 3) + (Cells(4, 3) * (Cells(i + 4, 10) - Cells(12,
3)) / 2) * Cells(3, 6) + ((Cells(6, 3) * Cells(12, 3) / 2 * Cells(4, 6))))
    Next i
End Sub

Private Sub fillCells3()
    Application.Volatile
    Dim i As Integer
    For i = 1 To 9
    'Cells(i + 4, 9) = i
    Cells(i + 4, 10) = Cells(10, 3) * i * 0.2
    Cells(i + 4, 11) = Cells(4, 3)

    FR = 1 - ((Cells(4, 3) * Cells(10, 3)) / (Cells(6, 3) * Cells(3, 3)))
    Cells(i + 4, 12) = (1 - FR) * Cells(6, 3) * Cells(3, 3) / Cells(i + 4, 10)
    Next i
End Sub

'--- obnovení grafu ---
Private Sub refreshFillCells()
    Call fillCells
    Call actualize
End Sub

'--- obnovení dvou grafu ---
Private Sub refreshFillCells2()
    Call fillCells
    Call fillCells2
    Call actualize
End Sub

Private Sub refreshFillCells3()
    Call fillCells3
    Call actualize
End Sub

Private Sub refreshFillCells4()
    Call fillCells4
    Call actualize
```

```
End Sub

Private Sub actualize()
    i = MsgBox("Úspěšně aktualizováno.", vbOKOnly + vbInfomation)
End Sub



'--- MODEL EOQ bez povoleného nedostatku ---

Function EOQ(Poptavka, Jednotkové_pořizovací_náklady, Jednotkové_skladovací_náklady)
    EOQ    =    Application.WorksheetFunction.Power(((2    *    Poptavka    *    Jednotkové_pořizovací_náklady)    /
Jednotkové_skladovací_náklady), 1 / 2)
End Function

Function TC_EOQ(Poptavka, Jednotkové_pořizovací_náklady, Jednotkové_skladovací_náklady)
    TC_EOQ    =    Application.WorksheetFunction.Power(2    *    Poptavka    *    Jednotkové_pořizovací_náklady    *
Jednotkové_skladovací_náklady, 1 / 2)
End Function

'--- MODEL EOQ s povoleným nedostatkem ---

Function EOQ2(Poptavka, Jednotkové_pořizovací_náklady, Jednotkové_skladovací_náklady, Jednotkové_náklady_neuspokojení)
    EOQ2    =    Application.WorksheetFunction.Power(((2    *    Poptavka    *    Jednotkové_pořizovací_náklady)    /
Jednotkové_skladovací_náklady),  1  /  2)  *  Application.WorksheetFunction.Power(((Jednotkové_skladovací_náklady  +
Jednotkové_náklady_neuspokojení) / Jednotkové_náklady_neuspokojení), 1 / 2)
End Function

Function S_EOQ2(Q, Jednotkové_skladovací_náklady, Jednotkové_náklady_neuspokojení) 'množství neuspokojené poptávky
    S_EOQ2 = Q * Jednotkové_skladovací_náklady / (Jednotkové_skladovací_náklady + Jednotkové_náklady_neuspokojení)
End Function

Function           TC_EOQ2(Poptavka,           Jednotkové_pořizovací_náklady,           Jednotkové_skladovací_náklady,
Jednotkové_náklady_neuspokojení)
    TC_EOQ2    =    Application.WorksheetFunction.Power((2    *    Poptavka    *    Jednotkové_pořizovací_náklady    *
Jednotkové_skladovací_náklady),  1  /  2)  *  (Application.WorksheetFunction.Power((Jednotkové_náklady_neuspokojení  /
(Jednotkové_náklady_neuspokojení + Jednotkové_skladovací_náklady)), 1 / 2))
End Function

'--- MODEL produkce a spotřeby ---

Function POQ(Poptavka, Intenzita_produkce, Intenzita_spotřeby, Náklady_výrobní_dávky, Jednotkové_skladovací_náklady)
    POQ = Application.WorksheetFunction.Power(((2 * Poptavka * Náklady_výrobní_dávky) / Jednotkové_skladovací_náklady),
1 / 2) * Application.WorksheetFunction.Power((Intenzita_produkce / (Intenzita_produkce - Intenzita_spotřeby)), 1 / 2)
End Function

Function TC_POQ(Poptavka, Intenzita_produkce, Intenzita_spotřeby, Náklady_výrobní_dávky, Jednotkové_skladovací_náklady)
    TC_POQ     =     Application.WorksheetFunction.Power((2     *     Poptavka     *     Náklady_výrobní_dávky     *
Jednotkové_skladovací_náklady), 1 / 2) * Application.WorksheetFunction.Power(((Intenzita_produkce - Intenzita_spotřeby)
/ Intenzita_produkce), 1 / 2)
End Function

'--- MODEL se stochastickou poptávkou - marginální přístup (RQM)---

Function RQM(Poptavka_odhad, Jednotkové_pořizovací_náklady, Jednotkové_skladovací_náklady)
    RQM    =    Application.WorksheetFunction.Power(((2    *    Poptavka_odhad    *    Jednotkové_pořizovací_náklady)    /
Jednotkové_skladovací_náklady), 1 / 2)
End Function

Function    RQM_Reorderpoint(Jednotkové_skladovací_náklady,    Jednotkové_náklady_neuspokojení,    Optimální_množství,
Poptavka_odhad, Sigma_M, Střední_hodnota_poptávky)
    FR = 1 - ((Jednotkové_skladovací_náklady * Optimální_množství) / (Jednotkové_náklady_neuspokojení * Poptavka_odhad))
    Z = Application.WorksheetFunction.Norm_S_Inv(FR)
    w = Z * Sigma_M
    RQM_Reorderpoint = Střední_hodnota_poptávky + w
End Function

Function           TC_RQM(Jednotkové_pořizovací_náklady,           Jednotkové_náklady_neuspokojení,           Sigma_M,
Jednotkové_skladovací_náklady, Optimální_množství, Poptavka_odhad, Střední_hodnota_poptávky, Bod_znovuobjednávky)
    FR = 1 - ((Jednotkové_skladovací_náklady * Optimální_množství) / (Jednotkové_náklady_neuspokojení * Poptavka_odhad))
    Z = Application.WorksheetFunction.Norm_S_Inv(FR)
    NZ = Application.WorksheetFunction.Norm_S_Dist(Z, False)
    TC_RQM = (Jednotkové_pořizovací_náklady + (Jednotkové_náklady_neuspokojení * Sigma_M * NZ * (Poptavka_odhad /
Optimální_množství)))  +  (((Optimální_množství  /  2)  +  (Bod_znovuobjednávky  -  Střední_hodnota_poptávky))  *
Jednotkové_skladovací_náklady)
End Function

'--- MODEL se stochastickou poptávkou - úroveň objednávky(RQL)---

Function RQL(Poptavka_odhad, Jednotkové_pořizovací_náklady, Jednotkové_skladovací_náklady)
    RQL    =    Application.WorksheetFunction.Power(((2    *    Poptavka_odhad    *    Jednotkové_pořizovací_náklady)    /
Jednotkové_skladovací_náklady), 1 / 2)
End Function
```

```
Function    RQL_Reorderpoint(Jednotkové_skladovací_náklady,    Jednotkové_náklady_neuspokojení,    Optimální_množství,
Poptavka_odhad, Sigma_M, Střední_hodnota_poptávky, Úroveň_objednávky)
    Z = Application.WorksheetFunction.Norm_S_Inv(Úroveň_objednávky / 100)
    w = Z * Sigma_M
    RQL_Reorderpoint = Střední_hodnota_poptávky + w
End Function

Function           TC_RQL(Jednotkové_pořizovací_náklady,           Jednotkové_náklady_neuspokojení,           Sigma_M,
Jednotkové_skladovací_náklady,  Optimální_množství,  Poptavka_odhad,  Střední_hodnota_poptávky,  Bod_znovuobjednávky,
Úroveň_objednávky)
    Z = Application.WorksheetFunction.Norm_S_Inv(Úroveň_objednávky / 100)
    NZ = Application.WorksheetFunction.Norm_S_Dist(Z, False)
    TC_RQL = (Jednotkové_pořizovací_náklady + (Jednotkové_náklady_neuspokojení * Sigma_M * NZ * (Poptavka_odhad /
Optimální_množství)))  +  (((Optimální_množství  /  2)  +  (Bod_znovuobjednávky  -  Střední_hodnota_poptávky))  *
Jednotkové_skladovací_náklady)
End Function

'--- MODEL se stochastickou poptávkou - jednorázově (RQT)---

Function RQT(c1, c2, Střední_hodnota, Sm_odchylka)
    gama = c2 / (c1 + c2)
    Z = Application.WorksheetFunction.Norm_S_Inv(gama)
    RQT = Střední_hodnota + Z * Sm_odchylka
End Function

'Option Explicit

Private Sub show_menu()
    Load MenuFunkci
    MenuFunkci.Show
End Sub
```

8.2 **Příloha č.2 – Ukázka zdrojového kódu metody EOQ**

```
Private Sub OkButton_Click()
    'Definice pole pro naplnění proměnnými z formuláře
    Dim promenne(15) As String
    Dim sheetName As String
    sheetName = ActiveSheet.Name
    'Naplnění pole
    promenne(1) = Me.TextBox1.Value
    promenne(2) = Me.TextBox2.Value
    promenne(3) = Me.TextBox3.Value
    promenne(4) = Me.TextBox4.Value
    promenne(5) = Me.TextBox5.Value
    'Založení nového listu pro výsledky, formátování sloupce s výsledky
    Sheets.Add(After:=Sheets(Sheets.Count)).Name = "Result " & Sheets.Count & " - EOQ"
    ActiveWindow.DisplayGridlines = False
    'Naplnění buněk hodnotami z textových polí formuláře
    ActiveSheet.Range("C3") = promenne(1)
    ActiveSheet.Range("C4") = promenne(2)
    ActiveSheet.Range("C5") = promenne(3)
    ActiveSheet.Range("C6") = promenne(4)
    ActiveSheet.Range("C7") = promenne(5)
    'Kontrola vstupu
    Call Numeric_Test
    'Naplnění popisných polí
    Call popisky
    'Vložení funkcí napojených na buňky s hodnotami
    ActiveSheet.Range("C10").Formula = "=Funkce.EOQ(C3, C5, C4)"
    ActiveSheet.Range("C12").Formula = "=Funkce.TC_EOQ(C3, C5, C4)"
    ActiveSheet.Range("C13").Formula = "=Funkce.TC_EOQ(C3, C5, C4)+ C3 * C6"
    ActiveSheet.Range("C15") = "=C3/C10"
    ActiveSheet.Range("C16") = "=12/C15"
    ActiveSheet.Range("C18") = "=MOD(C3/12*C7,C10)"
    'Formátování obsahu buněk, žluté (6) pro editaci, zelené (4) pro výsledky
    Range("C3:C7").Interior.ColorIndex = 6
    Range("C10").Interior.ColorIndex = 4
    Range("C12:C13").Interior.ColorIndex = 4
    Range("C15:C16").Interior.ColorIndex = 4
    Range("C18").Interior.ColorIndex = 4
    'Volání funkce pro změnu šířky sloupce B
    Call AutoFitColumns
    'Vložení grafu
    Application.Run "Funkce.fillCells"

    Range("J9:N9").Interior.ColorIndex = 4
    Range("J4:N14").Select
    Selection.NumberFormat = "0"
    ActiveSheet.Shapes.AddChart2(332, xlLineMarkers).Select
    With ActiveChart.Parent
```

```
         .Width = 450
         .Height = 240
         .Top = Range("I3").Top
         .Left = Range("I3").Left
     End With
    ActiveChart.SetSourceData Source:=Range("$K$4:$M$13")
    ActiveChart.ChartTitle.Text = "EOQ"
    ActiveChart.Axes(xlCategory, xlPrimary).HasTitle = True
    ActiveChart.Axes(xlCategory, xlPrimary).AxisTitle.Characters.Text = "Optimální velikost objednávky"
    ActiveChart.Axes(xlValue, xlPrimary).HasTitle = True
    ActiveChart.Axes(xlValue, xlPrimary).AxisTitle.Characters.Text = "Celkové náklady"
    ActiveChart.FullSeriesCollection(1).XValues = "='" & ActiveSheet.Name & "'!$J$5:$J$13"
    'Přidání tlačítka na aktualizaci grafu
    ActiveSheet.Buttons.Add(64, 340, 111, 35).Select
    Selection.Caption = "Refresh button"
    Selection.OnAction = "refreshFillCells"
    Selection.Characters.Text = "Obnovit graf"
    Range("A1").Select
    'Zavření formuláře
    Unload Me
End Sub

Private Sub AutoFitColumns()
    'Rozšíření sloupce B kvůli popisům
    ThisWorkbook.ActiveSheet.Columns("B:B").EntireColumn.AutoFit
    ThisWorkbook.ActiveSheet.Columns("C:C").EntireColumn.NumberFormat = "General"
End Sub

Private Sub ClearButton_Click()
    'Funkce pro vyčištění formuláře
    Dim Ctrl As Control
    For Each Ctrl In Me.Controls
        If TypeName(Ctrl) = "TextBox" Then
            Ctrl.Text = ""
        End If
    Next Ctrl
End Sub

Private Sub CancelButton_Click()
    'Funkce pro návrat do předchozího formuláře
    EOQ.Hide
    MenuFunkci.Show
End Sub

Sub Numeric_Test()

    Dim Ncell As Range
    Dim IsNotNumber As Boolean

    For Each Ncell In Range("C3:C5")
        If Not IsNumeric(Ncell) Then
            IsNotNumber = True
        End If
        If IsEmpty(Ncell) Then
            Ncell = 1
            MsgBox ("Prázdný vstup, doplněná hodnota 1.")
        End If
    Next Ncell

    For Each Ncell In Range("C6:C7")
        If Not IsNumeric(Ncell) Then
            IsNotNumber = True
        End If
        If IsEmpty(Ncell) Then
            Ncell = 0
        End If
    Next Ncell

    Select Case IsNotNumber
    Case True
        MsgBox ("Alespoň jeden vstup není numerický.")
        End
    End Select

End Sub

Private Sub popisky()
    ActiveSheet.Range("A1") = "Pro okamžitou aktualizaci výsledků editujte žlutě označené buňky. Pro aktualizaci
grafu použijte tlačítko."
    ActiveSheet.Range("B3") = "Poptavka: "
    ActiveSheet.Range("B4") = "Jednotkové skladovací náklady: "
    ActiveSheet.Range("B5") = "Jednotkové pořizovací náklady: "
    ActiveSheet.Range("B6") = "Cena jednotky: "
    ActiveSheet.Range("B7") = "Pořizovací lhůta (měs.): "
    ActiveSheet.Range("B10") = "Optimální velikost objednávky: "
```

```
    ActiveSheet.Range("B12") = "Optimální celkové náklady: "
    ActiveSheet.Range("B13") = "Celkové náklady včetně zboží: "
    ActiveSheet.Range("B15") = "Interval objednávky (krát/rok): "
    ActiveSheet.Range("B16") = "Interval objednávky (jednou za): "
    ActiveSheet.Range("B18") = "Bod znovuobjednávky: "
    ActiveSheet.Range("D16") = "měsíce"
    ActiveSheet.Range("J4") = "Objednáváné množství"
    ActiveSheet.Range("K4") = "Náklady objednání"
    ActiveSheet.Range("L4") = "Náklady skladování"
    ActiveSheet.Range("M4") = "Celkové náklady"
    ActiveSheet.Range("N4") = "Celkové náklady včetně zboží"
End Sub
```