Univerzita Pardubice

Dopravní fakulta Jana Pernera

Analytické a simulační přístupy k řešení úloh teorie hromadné obsluhy

Bc. František Daňhel

Diplomová práce

2019

Univerzita Pardubice
Dopravní fakulta Jana Pernera
Akademický rok: **2018/2019**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. František Daňhel**
Osobní číslo:	**D17330**
Studijní program:	**N3708 Dopravní inženýrství a spoje**
Studijní obor:	**Aplikovaná informatika v dopravě**
Název tématu:	**Analytické a simulační přístupy k řešení úloh teorie hromadné obsluhy**
Zadávající katedra:	**Katedra informatiky v dopravě**

Zásady pro vypracování:

Cílem diplomové práce je srovnání analytického a simulačního přístupu při řešení základních modelů teorie hromadné obsluhy. V teoretické části práce budou popsány základní poznatky a principy z teorie hromadné obsluhy, zejména základní používané modely v této teorii (model M|M|1, M|M|n, M|D|1, M|G|1 a další) včetně základních parametrů typu "počet zákazníků v systému", "počet zákazníků ve frontě", "pravděpodobnost čekání" atd. V praktické části práce bude ve vhodném nástroji provedena simulace základních modelů včetně řešených parametrů a výsledky této simulace budou srovnány s analytickou metodou výpočtu těchto parametrů.

Rozsah grafických prací:

Rozsah pracovní zprávy: **40 normostran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

LINDA, Bohdan. Stochastické metody operačního výzkumu. Bratislava: Statis, 2004. ISBN 80-85659-33-6.
ZIMOLA, Bedřich. Operační výzkum. Vyd. 4., nezm. Zlín: Univerzita Tomáše Bati, 2004. ISBN 80-7318-208-4.
JABLONSKÝ, Josef. Operační výzkum: kvantitativní modely pro ekonomické rozhodování. 2. vyd. Praha: Professional Publishing, 2002. ISBN 80-86419-42-8.
PEŠKO, Štefan. Teória hromadnej obsluhy: Prednášky pre AM [online]. 2001. Dostupné z: http://frcatel.fri.uniza.sk/users/pesko/OA2/oa2m.pdf
GROSS, Donald a Carl M HARRIS. Fundamentals of queueing theory. 2nd ed. New York: Wiley, 1985. ISBN 0-471-89067-7.

Vedoucí diplomové práce: **Ing. Ondřej Míča**
Katedra informatiky v dopravě

Datum zadání diplomové práce: **11. prosince 2018**

Termín odevzdání diplomové práce: **24. května 2019**

L.S.

doc. Ing. Libor Švadlenka, Ph.D.
děkan

doc. Ing. Karel Greiner, Ph.D.
vedoucí katedry

V Pardubicích dne 11. prosince 2018

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 22. 5. 2019

Bc. František Daňhel

Poděkování

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Ondřeji Míčovi za jeho čas, cenné rady a připomínky, které mi byly poskytnuty a přispěly ke zlepšení a zkvalitnění této práce.

ANOTACE

Práce se zabývá systémy hromadné obsluhy a srovnává analytický a simulační přístup jejich řešení. Seznamuje s teorií hromadné obsluhy a uvádí vzorce potřebné pro analytické řešení běžných modelů. Popisuje návrh a implementaci simulačního softwaru, který byl v rámci práce vyvinut v programovacím jazyce Java. Na konkrétních příkladech běžných modelů jsou porovnány výsledky analytického a simulačního řešení.

KLÍČOVÁ SLOVA

systémy hromadné obsluhy, teorie hromadné obsluhy, počítačová simulace

TITLE

Analytical and simulation approaches to solving the problems of queueing theory

ANNOTATION

The thesis deals with queueing systems and it compares analytical and simulation approach to solve them. It introduces queueing theory and presents formulas required to analyze basic models. It describes design and implementation of the simulation software, which was developed during the work in the Java programming language. Results of the analytical and simulation solution are compared in concrete examples of the basic models.

KEYWORDS

queueing systems, queueing theory, computer simulation

OBSAH

Seznam obrázků **9**

Seznam tabulek **10**

Seznam zkratek **11**

Úvod **12**

1 Teorie hromadné obsluhy **13**

1.1 Základní prvky systémů hromadné obsluhy 15

1.1.1 Zdroj a příchod zákazníků 16

1.1.2 Fronta 17

1.1.3 Linka obsluhy 19

1.1.4 Síť obslužných linek 19

1.2 Kendallova klasifikace 21

1.3 Analýza systémů hromadné obsluhy 23

1.3.1 Řešení systémů hromadné obsluhy 23

1.3.2 Charakteristiky systémů hromadné obsluhy 24

1.3.3 Optimalizace systémů hromadné obsluhy 26

2 Analytické řešení **27**

2.1 Náhodné procesy 27

2.1.1 Markovovy procesy 28

2.1.2 Grafová reprezentace 33

2.1.3 Poissonův náhodný proces 35

2.2 Systém M|M|1 37

2.3 Systém M|M|c 42

2.4 Systém M|M|c|K 44

2.5 Systém M|D|1 46

2.6 Systém M|G|1 47

3 Simulační řešení **49**

3.1 Modelování a simulace 49

3.1.1 Generování náhodných veličin 50

3.1.2 Diskrétní simulace 51

3.2 Analýza a specifikace požadavků 51

3.2.1 Obecné požadavky na aplikaci 51

3.2.2 Požadavky kladené na počítačový model SHO 52

3.2.3 Požadavky kladené na GUI 52

3.3 Návrh aplikace 53

3.4 Modul simulačního modelu 53

3.4.1 Návrh tříd a implementace 53

3.4.2 Ukázka použití modulu simulačního modelu 61

3.5 Modul statistiky 63

3.6 Modul GUI 64

3.6.1 Návrh GUI 64

3.6.2 Implementace GUI 66

4 Srovnání analytického a simulačního řešení **69**

4.1 Model M|M|1 71

4.2 Model M|M|c 72

4.3 Model M|M|c|K 73

4.4 Model M|D|1 74

4.5 Model M|G|1 75

Závěr **77**

Použitá literatura **79**

Seznam příloh **81**

Příloha A **82**

SEZNAM OBRÁZKŮ

1 Schématický nákres systému hromadné obsluhy 14
2 Paralelní uspořádání linek obsluhy s vlastními frontami 20
3 Sériové uspořádání linek obsluhy s vlastními frontami 20
4 Sériové uspořádání linek obsluhy s jedinou frontou 21
5 Grafické znázornění pravděpodobností přechodu pomocí grafu 33
6 Grafické znázornění intenzity pravděpodobností přechodu pomocí grafu 34
7 Přechodový graf Poissonova náhodného procesu 35
8 Pravděpodobnosti stavů Poissonova procesu s parametrem $\lambda = 4$ 36
9 Funkce hustoty pravděpodobnosti exponenciálního rozdělení s parametrem $\lambda = 4$ 38
10 Schématické znázornění systému M|M|1 38
11 Přechodový graf intenzit pravděpodobností systému M|M|1 40
12 Schématické znázornění systému M|M|c 42
13 Přechodový graf intenzit pravděpodobností systému M|M|c 43
14 Diagram vybraných tříd z balíčku qs.model 54
15 Vizualizace třídy Component 58
16 Vizualizace třídy Simulator 61
17 Návrh grafického uživatelského rozhraní 65
18 Analytické řešení pomocí tabulkového procesoru 69
19 Simulační řešení M|M|3 pomocí simulačního softwaru 70

SEZNAM TABULEK

1 Symboly používané v Kendalově klasifikaci 22
2 Označení charakteristik použité v této práci 25
3 Analytické a simulační řešení modelu M|M|1 71
4 Analytické a simulační řešení modelu M|M|3 72
5 Analytické a simulační řešení modelu M|M|2|6 73
6 Analytické a simulační řešení modelu M|D|1 74
7 Analytické a simulační řešení modelu M|G|1 76

SEZNAM ZKRATEK

API	Application Program Interface (aplikační programové rozhraní)
CLI	Command Line Interface (rozhraní příkazové řádky)
FIFO	First In, First Out (řádný režim fronty; první příchozí do fronty odchází z fronty první)
GUI	Graphical User Interface (grafické uživatelské rozhraní)
LIFO	Last In, First Out (inverzní režim fronty; poslední příchozí do fronty odchází z fronty první)
PRI	Priority (prioritní režim fronty; z fronty odchází prvky dle přiřazených priorit)
SHO	systém hromadné obsluhy
SIRO	Service In Random Order (náhodný režim fronty; odcházející prvek je z fronty náhodně vybrán)
THO	teorie hromadné obsluhy

ÚVOD

V každodenním životě se můžeme potkat s mnoha situacemi, kdy určitý požadavek nemůže být vyřízen ihned, ale je třeba, aby vyčkal. Může se jednat o zákazníky, kteří čekají u přepážek v bance, na poště nebo na úřadě. Stejně tak, se může jednat o frontu síťových paketů, které čekají na zpracování určitým procesem (vláknem) na serveru apod.

Ve zmíněných a dalších situacích lze využít teorie a poznatků, které k této problematice již existují. Jedná se o teorii hromadné obsluhy, která zkoumá a analyzuje zmíněné systémy na základě teorie pravděpodobnosti. Jiným způsobem je vytvoření počítačového modelu a prováděním různých experimentů, při kterých jsou měřeny charakteristické vlastnosti systémů. Cílem práce je přiblížit a porovnat oba zmíněné přístupy.

V první kapitole práce je čtenář seznámen se samotnou teorií hromadné obsluhy, kde jsou popsány základní prvky systémů, Kendallova klasifikace a jsou zde uvedeny charakteristické veličiny získané při řešení systémů hromadné obsluhy.

Druhá kapitola se zabývá analytickým řešením. Čtenář je seznámen s existující teorií náhodných a Markovových procesů, na jejím základě jsou odvozeny vzorce pro získání charakteristik systémů. V kapitole jsou představeny běžné modely systémů, včetně jejich analytického řešení.

Třetí kapitola seznamuje s druhým způsobem řešení systémů hromadné obsluhy, kterým je počítačová simulace. Kromě uvedení základních pojmů a principů, kapitola popisuje a specifikuje vývoj softwaru, který umožňuje vytváření a zkoumání různých systémů hromadné obsluhy.

V poslední čtvrté kapitole čtenář nachází srovnání analytického a simulačního řešení systémů hromadné obsluhy. Pro simulační řešení je využit software, který byl speciálně vyvinut v rámci této diplomové práce.

1 TEORIE HROMADNÉ OBSLUHY

Teorie hromadné obsluhy (THO), někdy označovaná též jako teorie front, zkoumá kvantitativně procesy hromadného charakteru, které jsou charakterizovány vznikem požadavků spojených s určitými jednotkami vyžadujícími obsluhu, tj. provedení jistých operací. Vlivem omezené kapacity obsluhy může docházet k hromadění (čekání) jednotek s následným uspokojením požadavku nebo odmítnutím obsluhy. [2, str. 86]

Systémem hromadné obsluhy, lze chápat zařízení, které se skládá z následujících prvků:

- zákazník,
- fronta,
- linka obsluhy.

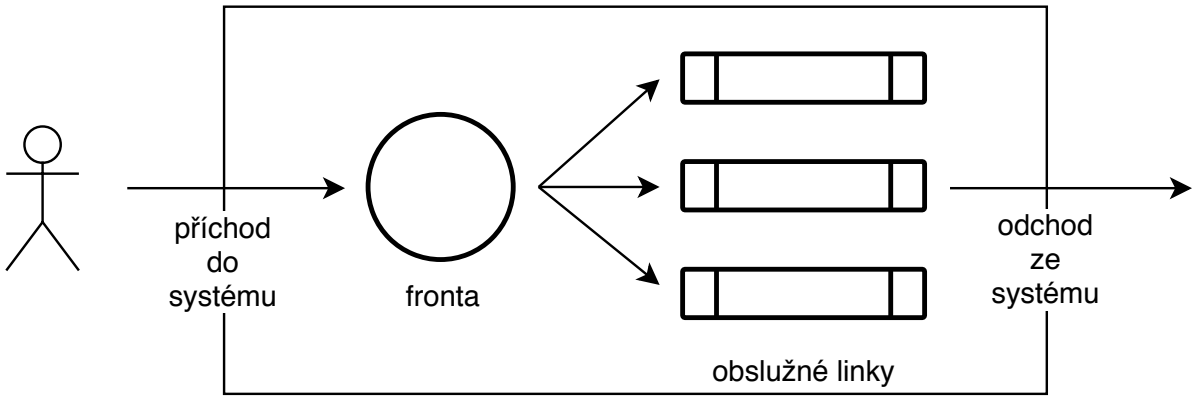
Linka obsluhy nabízí určitou činnost, či proces, který poptávají **zákazníci**. Zákazníci za tímto účelem přichází do systému. Samotná činnost prováděná linkou obsluhy, kdy je obsluhován konkrétní zákazník, ovšem trvá určitý časový interval. Během tohoto časového okamžiku může přijít do systému další zákazník, který vyžaduje obsluhu. Vzhledem k tomu, že na lince obsluhy se lze zpravidla věnovat pouze jednomu zákazníkovi současně, tak mohou nastat 2 situace:

1. V systému existuje jiná linka obsluhy, která je v nečinnosti a tato se pak ujme obsluhy dotyčného zákazníka.
2. V systému jsou všechny linky obsluhy právě zaneprázdněny a zákazník musí počkat, než bude některá z linek obsluhy volná. Za tímto účelem si stoupne do **fronty**, kde vyčkává do doby, než se na něj dostane řada a bude obsloužen.

Jakmile je na lince obsluhy ukončena obsluha zákazníka, tak linka obsluhy začne obsluhovat dalšího zákazníka z fronty, pokud fronta není prázdná. Pokud aktuálně ve frontě žádný zákazník na obsluhu nečeká, tak linka obsluhy přestane být aktivní a vyčkává na příchod nového zákazníka, kterého by mohla obsloužit. [1, str. 21]

Obrázek 1 znázorňuje možné grafické pojetí systému hromadné obsluhy. Ilustrace vyobrazuje systém se třemi linkami obsluhy a jednou společnou frontou. Linky jsou uspořádány paralelně a jsou vzájemně plně zastupitelné — pro zákazníka je tedy zcela nepodstatné, kterou linkou obsluhy bude obsloužen.

Systémy hromadné obsluhy se staly běžnou součástí našich každodenních životů. Setkáváme se s nimi v mnoha různých formách a v různých situacích. Prakticky se může



Obrázek 1: Schématický nákres systému hromadné obsluhy. Zpracování autora inspirované v [3, str. 263]

jednat o jakékoliv místo, kde se vytváří fronty zákazníků, či požadavků. Je třeba poznamenat, že pod pojmem *zákazníci* se ve skutečnosti nemusí skrývat pouze lidé, kteří někde na něco čekají, ale může se jednat např. o nevyřízené smlouvy, příchozí hovory apod. Existuje nespočetné množství příkladů systému hromadné obsluhy, mezi kterými mohou být následující případy, či situace: [3, str. 264]

- *Ordinace lékaře*: Zákazníky jsou přicházející pacienti, kteří tvoří frontu v čekárně. Linkou obsluhy je lékař nebo lékaři, kteří pacienty postupně přijímají a vyšetřují, popř. konzultují jejich problémy.
- *Banka*: Zákazníky jsou klienti banky, kteří přicházejí na pobočku. Linkami obsluhy jsou jednotlivé přepážky s pracovníky banky (za předpokladu, že všechny přepážky jsou rovnocenné a obsluhují všechny typy požadavků klientů).
- *Samoobsluha*: V samoobsluze jsou zákazníky lidé, přicházející do obchodu. Jako linky obsluhy lze vnímat buď jednotlivé pokladny, u kterých se tvoří fronty nebo nákupní vozíky, kterých je omezený počet a např. při velkém množství zákazníků v obchodu je nutné na ně čekat.
- *Výrobní linka*: Zákazníky systému nejsou, narozdíl od předcházejících příkladů, živé lidské bytosti, ale za zákazníky se považují jednotlivé výrobky postupující výrobním procesem. Linkami obsluhy mohou být např. jednotlivá místa na výrobní lince, ve kterých jsou výrobky dále opracovávány.
- *Dopravní systém*: Zákazníky jsou komplety pohybující se po dopravní síti. Linky obsluhy tvoří jednotlivé křižovatky, které mohou být řízeny světelným signalizačním

zařízením. Příchodem zákazníka do systému se považuje příjezd kompletu k dané křižovatce.

- *Čerpací stanice*: Zákazníky jsou vozidla přijíždějící na čerpací stanici. Linkami obsluhy jsou jednotlivé stojany s pohonnými hmotami.
- *Nádraží*: Zákazníky tvoří cestující, kteří přichází za účelem nákupu jízdenky. Linkami obsluhy jsou jednotlivé pokladní přepážky, kde se prodávají jízdenky.
- *Pojišťovna*: Zákazníky nejsou opět lidé, ale tvoří je nevyřízené pojistné případy. Linkami obsluhy jsou jednotliví úředníci, kteří se zmiňovanými pojistnými případy zabývají a řeší je.
- *Lyžařská sjezdovka*: Zákazníky tvoří jednotlivý lyžaři (či snowboardisté), kteří se chtějí dostat vlekem do horní části sjezdovky. Linky obsluhy tvoří jednotlivá místa na vleku (např. kotvy).
- *Letiště*: Zákazníky tvoří přilétající letadla, která se chystají na přistání. Linkami obsluhy jsou jednotlivé přistávající dráhy.
- *Strojírenská výroba*: Zákazníky tvoří jednotlivé stroje, které je třeba seřídit. Linkami obsluhy jsou jednotliví seřizovači, kteří stroje ve firmě seřizují.
- *Počítačová síť*: Zákazníky tvoří zadané požadavky na tisk dokumentů z jednotlivých počítačů. Linky obsluhy tvoří tiskárny zapojené do počítačové sítě.
- *Mycí stanice*: Zákazníky tvoří vozidla přijíždějící k myčce vozidel. Linky obsluhy tvoří jednotlivé mycí linky.
- *Seřaďovací nádraží*: Zákazníky tvoří vozy čekající na rozřazení. Linkami obsluhy jsou svážné pahrbky, na kterých může ve stanici probíhat současné rozřazování.

1.1 Základní prvky systémů hromadné obsluhy

Jak je zřejmé z předchozích příkladů, tak systémy hromadné obsluhy mohou nabývat různých struktur s rozdílnými charakteristikami. V systémech hromadné obsluhy proto rozlišujeme základní klíčové prvky, ze kterých se skládají. U těchto prvků se následně určují jejich vlastnosti a charakteristiky. Na základě charakteru dílčích prvků, lze poté charakterizovat a řešit celý systém.

1.1.1 Zdroj a příchod zákazníků

Do systému hromadné obsluhy přicházejí zákazníci, kteří požadují obsluhu. Z pohledu modelu systému lze vnímat dva druhy zdroje zákazníků [3, str. 265]:

- konečný,
- nekonečný.

V případě *konečného zdroje* zákazníků se v modelu pracuje s určitým konečným počtem zákazníků. Tento typ zdroje se používá v modelech, kde je počet zákazníků v systémů omezený a relativně malý. Příkladem může být několik málo desítek strojů ve výrobní hale, které je třeba udržovat a opravovat.

Naproti tomu *nekonečný zdroj* zákazníků, jak už z názvu napovídá, není omezen žádným konečným počtem zákazníků a v modelu se uvažuje, že z tohoto zdroje do systému přichází stále noví zákazníci. Zdroj je často považován za nekonečný, i když ve skutečnosti se může jednat o omezenou množinu zákazníků. Pokud ovšem je počet zákazníků dostatečně velký (stovky až tisíce), tak lze považovat zdroj za nekonečný. Příkladem jsou příchody pacientů k lékaři, či klientů do banky.

Důležitým údajem souvisejícím s příchodem zákazníků do systému je **intenzita příchodů** nebo **interval mezi příchody** [3, str. 265]. Intenzita příchodů udává počet příchodů zákazníku do systému za určitou časovou jednotku. Interval mezi příchody (resp. střední hodnota intervalu mezi příchody) udává čas mezi dvěma bezprostředně následujícími příchody zákazníků. Obě veličiny jsou navzájem velmi silně spjaty, protože jedna je převrácenou hodnotou druhé a naopak. Stačí nám tedy znát pouze jednu z veličin. Matematicky lze vztah mezi veličinami vyjádřit následovně:

$$\text{intenzita příchodů} = \frac{1}{\text{interval mezi příchody}} \tag{1}$$

Pokud je například intenzita příchodů 6 osob za hodinu, pak je interval mezi příchody $^1/_6$ hodiny, tedy 10 minut.

Výše uvedené veličiny (intenzitu příchodů a interval mezi příchody) lze rozdělit do dvou skupin:

- deterministické,
- stochastické.

Deterministické veličiny jsou charakterizovány přesnou definicí hodnoty, v tomto případě okamžiku příchodu, popř. intervalu mezi dvěma příchody. Příkladem determinis-

tických příchodů může být např. výrobní linka, kde vznikají výrobky v předem daných neměnných intervalech.

Naproti tomu *stochastické* veličiny jsou charakteristické právě jejich náhodností a není tedy dopředu zřejmé, kdy přesně nastane další příchod zákazníka. Z praktického pohledu jsou ovšem právě systémy s náhodností zajímavější a často se s nimi lze více přiblížit k realitě, kde zákazníci často nepřichází v přesně daných intervalech, ale z pohledu THO náhodně. I když systémy obsahují prvky náhodnosti, lze je řešit a zkoumat, pokud známe podrobnosti o rozdělení pravděpodobnosti příchodů zákazníků nebo doby obsluhy. Z pohledu THO je významné exponenciální rozdělení pravděpodobnosti, kterým se často aproximují intervaly příchodů zákazníků anebo doby obsluhy reálných systémů. [1, str. 21]

1.1.2 Fronta

Fronta tvoří v systému hromadné obsluhy významný prvek, kde jsou shlukovány zákazníci, čekající na obsluhu. Typicky bývá fronta předřazena jedné nebo více linkám obsluhy. Zákazník frontu využivá pouze v případě, kdy při jeho příchodu do systému není volná žádná linka obsluhy, která by jej obsloužila.

Frontu lze charakterizovat z několika hledisek — především se jedná o frontový režim, kapacitu fronty a způsob čekání a následné obsluhy zákazníků s vyšší prioritou.

Frontový režim určuje předpoklady o způsobu, v jakém pořadí zákazníci opouští frontu a vstupují do obsluhy. To má přímý vliv na dobu, kterou stráví jednotliví zákazníci ve frontě. Běžně užívané frontové režimy jsou [2, str. 87]:

- **FIFO** *(First In, First Out), řádný režim.* Jedná se o v životě běžný způsob tvorby fronty. Přicházející zákazník se řadí jako poslední na konec fronty. Z fronty vždy odchází zákazník na začátku fronty. Tedy ten, který přišel ze všech ostatních nejdříve a čeká ze všech ostatních ve frontě nejdéle. Režim je vůči zákazníkům „nejvíce spravedlivý“. Pro označení se někdy používá také zkratky FCFS (First Come, First Served). [5, str. 8]
- **LIFO** *(Last In, First Out), inverzní režim.* Zákazníci jsou ve frontě sice řazeny podle času příchodů, ovšem z fronty jako první odchází ten, který přišel do fronty jako poslední. V praxi tento režim fronty může být např. mezi výrobními linkami, kdy jsou výrobky odkládány na sebe (např. tabule skla). Ačkoliv je střední doba čekání ve frontě stejná jako v případě režimu FIFO, vyskytují se velké rozdíly mezi do-

bami čekání u jednotlivých zákazníků, což se projevuje rozdílným rozptylem oproti zmíněnému režimu FIFO. Režim je vůči zákazníkům „nejméně spravedlivý“. Pro označení se někdy používá také zkratky LCFS (Last Come, First Served). [5, str. 8]

- **SIRO** *(Service In Random Order).*[1] Zákazníci jsou z fronty vybíráni v náhodném pořadí. Tento typ režimu neříká žádné další informace o procesu náhodného výběru, lze tedy předpokládat jakékoliv rozdělení pravděpodobnosti.
- **PRI** *(Priority).* Tento režim předpokládá, že zákazníci mají přiřazenou různou prioritu. Frontu nejprve opouští zákazníci s nejvyšší prioritou (nejdůležitější) a poté postupně s prioritami nižšími.
- **GD** *(General Discipline).* Obecný řád fronty. Neexistují žádné předpoklady o způsobu řazení. Zákazníci tedy frontu mohou opouštět v libovolném pořadí.

Není-li dále uvedeno jinak, tak se v této práci předpokládá vždy s řádný frontovým režimem (FIFO).

Dalším kritériem, určujícím chování fronty a potažmo celého systému je **kapacita fronty**. Z pohledu omezenosti kapacity lze rozlišovat frontu:

- s ohraničením,
- bez ohraničení.

Fronta **s ohraničením** má omezenou kapacitu na určitý počet míst, který nesmí být překročen. Pokud je fronta plně obsazená a již není místo pro nově příchozího zákazníka, tak je zákazník odmítnut a zpravidla odchází ze systému neobsloužen. Speciálním případem je fronta, jejíž kapacita je nulová. V tomto případě zákazníci nemohou tvořit frontu a pokud příjde zákazník a všechny linky obsluhy jsou právě obsazeny, tak je zákazník odmítnut. O takovémto systému hovoříme jako o *systému s odmítáním.* [4, str. 7]

Pokud se jedná o frontu **bez ohraničení**, tak nedochází v systému k žádnému odmítání zákazníků. Délka fronty může být teoreticky nekonečná, ovšem v praxi není žádoucí, aby k takovýmto frontám docházelo a je často požadována *stabilizace systému.* [6, str. 75]

Charakter a výčet vlastností front, popsaný v odstavcích výše není konečný. Fronty mohou mít některé speciální rysy, jakými jsou např. časově proměnné priority zákazníků (např. závislé na již strávené době ve frontě/systému), technologií dané pevné pořadí zpracování prvků nebo omezenou dobu čekání zákazníka ve frontě (rezignace) atd.

[1] V literatuře [2, str. 87] uváděn také výraz „Selection In Random Order“

1.1.3 Linka obsluhy

Linka obsluhy je místo, které v systému hromadné obsluhy zastupuje určitý obslužný proces zákazníka. Z pohledu systémů hromadné obsluhy není podstatné, jakou konkrétní činnost linka obsluhy vykonává. Podstatné jsou charakteristiky, jak dlouho je zákazník obsluhován, zda je možné obsloužit více zákazníků hromadně nebo způsob přerušení obsluhy zákazníka, pokud do fronty přijde zákazník s vyšší prioritou.

Doba obsluhy může být (podobně jako u příchodů zákazníků): [2, str. 266]

- deterministická,
- stochastická.

Příkladem deterministické doby obsluhy může být vlek pro lyžaře, pokud za dobu obsluhy považujeme časový interval mezi přistavením dvou po sobě následujících kotev. Naproti tomu, pokud se budeme snažit vytvořit model lékařské ordinace, tak pravděpodobně budeme dobu obsluhy zákazníka (pacienta) modelovat jako strochastickou (náhodnou) veličinu s daným rozdělením pravděpodobnosti.

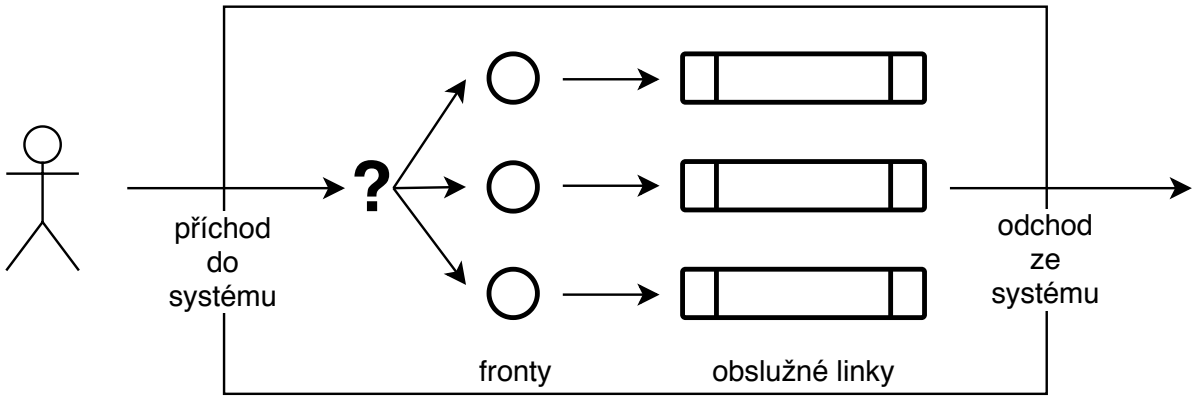
Podobně, jako u zdroje zákazníků souvisela intenzita příchodů s intervalem mezi příchody, tak i stejná nepřímá úměra platí mezi intenzitou a dobou obsluhy (resp. střední hodnotou doby obsluhy). Obecně lze tedy předpokládat, že: [3, str. 266]

$$\text{intenzita obsluhy} = \frac{1}{\text{doba obsluhy}} \tag{2}$$

1.1.4 Síť obslužných linek

Uspořádání a celkový počet linek obsluhy ovlivňuje fungování a výsledné charakteristiky celého systému hromadné obsluhy. Cílem zkoumání nemusí být jen hodnoty základních charakteristik systémů, ale může se hledat takové celkové uspořádání systému (zejména počet linek), aby provoz systému byl z určitého hlediska optimální (např. vytíženost linek, celkové ekonomické náklady apod.). [3, str. 279]

Nejjednodušší uspořádání má systém, který obsahuje jednu frontu a jednu linku obsluhy. Kombinací různých prvků (především však linek obsluhy) vzniká síť obslužných linek, která může mít různě složité fungování. U některých uspořádání existují pro hledání charakteristik systémů známá analytická řešení. Tam, kde neexistuje analytické řešení lze zvolit řešení pomocí simulace. Existují dvě specifická uspořádání sítě obslužných linek: paralelní a sériové. [2, str. 87]

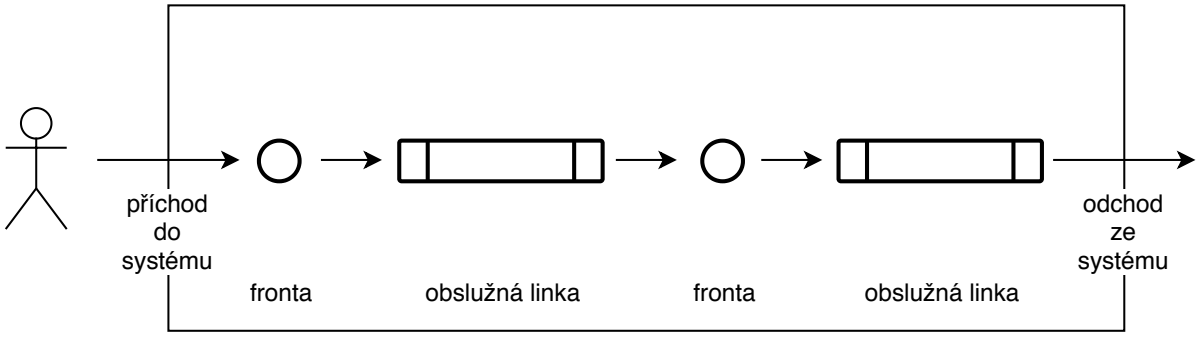


Obrázek 2: Paralelní uspořádání linek obsluhy s vlastními frontami. Zpracování autora.

Paralelní uspořádání je charakteristické několika linkami obsluhy, které jsou navzájem zastupitelné (zákazníkovi nezáleží na tom, kterou linkou bude obsloužen) a mají stejnou charakteristiku (doba obsluhy je na všech linkách stejného typu se stejnými parametry). Linky obsluhy mohou sdílet stejnou frontu zákazníků (obr. 1), nebo může mít každá linka svoji vlastní frontu (obr. 2).

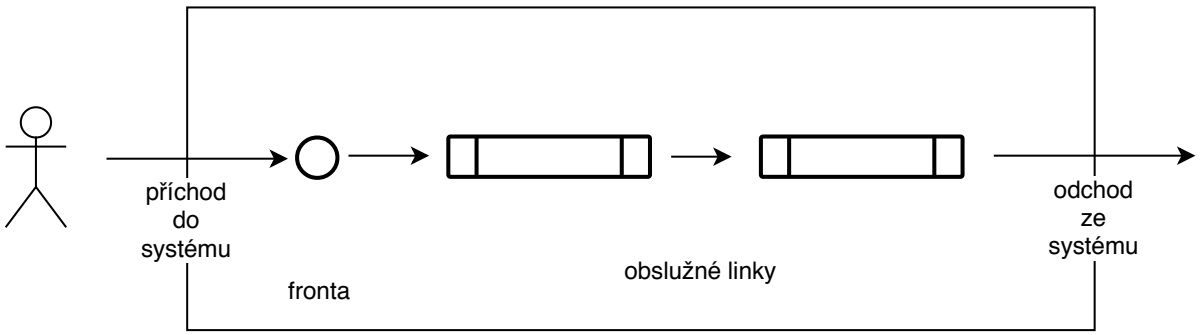
Dalším přístupem může být také nahrazení původního zdroje zákazníků tak, aby každá fronta měla svůj vlastní zdroj se sníženou (úměrně počtu front) intenzitou příchodů zákazníků.

Sériové uspořádání předpokládá zřetězení linek obsluhy za sebe. Zákazník je postupně obsluhován různými linkami obsluhy. Fronta může být buďto před každou linkou obsluhy (obr. 3) nebo pouze před první linkou obsluhy (obr. 4). Pokud je fronta před každou linkou, systém má tu výhodu, že zákazník po dokončení obsluhy linku ihned opustí a ta je volná pro další zákazníky. V opačném případě (tedy pokud fronta vzniká pouze před první linkou) mohou nastat po dokončení obsluhy zákazníka na dané lince dvě možnosti: [4, str. 75]



Obrázek 3: Sériové uspořádání linek obsluhy s vlastními frontami. Zpracování autora.

- *následující linka je volná* — zákazník stávající linku ihned opouští a začíná jeho obsluha na následující lince,
- *následující linka je obsazená* — zákazník setrvává na aktuální lince, kterou blokuje do doby než skončí obsluha předchozího zákazníka na následující lince.



Obrázek 4: Sériové uspořádání linek obsluhy s jedinou frontou. Zpracování autora.

1.2 Kendallova klasifikace

Pro roztřídění a jednotný systematický popis systémů hromadné obsluhy vytvořil anglický matematik David George Kendall na počátku 50. let 20. století[2] klasifikaci, která má ve zkratce zakódované základní informace o systému hromadné obsluhy. Původně byla používána ve formě tří uspořádaných znaků, ale postupně se ukázalo, že je to nedostatečné a klasifikace byla rozšířena na pět znaků. [7]

Původní tříznakovou notaci lze vyjádřit v následujícím tvaru (místo znaku svislé čáry se často používá také lomítko):

$$\boxed{\text{A}|\text{B}|\text{x}}$$

Na pozici jednotlivých symbolů (písmen A, B, x) jsou dosazovány konkrétní hodnoty (písmena nebo čísla) charakterizující daný aspekt systému hromadné obsluhy. Pozice mají následující význam:

- Na pozici **A** je písmeno udávající charakterizující typ a pravděpodobnostní rozdělení časového intervalu mezi jednotlivými příchody zákazníků do systému. Podrobnosti v tabulce 1.
- Na pozici **B** je písmeno udávající charakterizující typ a pravděpodobnostní rozdělení doby obsluhy na linkách obsluhy. Podrobnosti v tabulce 1.

[2]Konkrétní rok se napříč literaturou liší. V [7] je uveden rok 1951, zatímco v [4, str. 8] rok 1953.

- Pozice **x** obsahuje číslo, které udává počet paralelních linek v systému. Předpokládá se, že všechny linky jsou navzájem zastupitelné a mají stejné pravděpodobnostní rozdělení doby obsluhy, charakterizované v notaci na pozici *B*.

Tabulka 1: Symboly používané v Kendalově klasifikaci. Zdroj [1, 4]

Písmeno	**Dosazené na pozici A**	**Dosazené na pozici B**
M	Příchody zákazníků tvoří Poissonův proces, tzn. intervaly mezi příchody zákazníků mají exponenciální rozdělení pravděpodobnosti.	Doba obsluhy má exponenciální rozdělení pravděpodobnosti.
E_k	Intervaly mezi příchody zákazníků mají Erlangovo rozdělení pravděpodobnosti s parametry λ a k.	Doba obsluhy má Erlangovo rozdělení pravděpodobnosti s parametry λ a k.
H_k	Intervaly mezi příchody zákazníků mají hypererlangovo rozdělení pravděpodobnosti.	Doba obsluhy má hypererlangovo rozdělení pravděpodobnosti.
K_n	Intervaly mezi příchody zákazníků mají χ^2 rozdělení s n stupni volnosti.	Doba obsluhy má χ^2 rozdělení s n stupni volnosti.
D	Deterministické příchody zákazníků.	Konstantní doba obsluhy.
G	Intervaly mezi příchody zákazníků mají blíže nespecifikované rozdělení pravděpodobnosti, u kterého je ovšem známa střední hodnota a směrodatná odchylka.	Doba obsluhy má libovolné rozdělení pravděpodobnosti se známou střední hodnotou a royptylem.
GI	Příchody zákazníků tvoří rekurentní proces.	*(nepoužívá se)*
Y(k)	*(nepoužívá se)*	Doba obsluhy je závislá na stavu systému, tedy aktuálním počtu zákazníků v systému.

U původní notace ve formě tří uspořádaných znaků se postupně ale ukázalo, že je to nedostatečné a klasifikace byla rozšířena na pět znaků: [7]

$$\boxed{\text{A|B|x|y|z}}$$

Oproti tříznakové notaci přibyly pozice y a z, které následovně rozšiřují informace o systému:

- Pozice **y** obsahuje číslo (nebo symbol ∞) udávající kapacitní omezení systému. Pokud je v systému právě tolik zákazníků, jako je jeho kapacita, nově příchozí zákazníci jsou odmítání do chvíle, než se počet zákazníků v systému opět sníží. Některá literatura (např. [1, str. 24]) udává, že se nejedná o kapacitu systému, ale fronty (tedy nejsou započítána místa na linkách obsluhy). V této práci bude nadále uvažováno s kapacitou systému. Symbol ∞ znamená, že systém/fronta nemá omezenou kapacitu.
- Pozice **z** obsahuje zkratku označující řád fronty. Běžně se používá zkratek FIFO, LIFO, SIRO, PRI a GD. Význam zkratek je uveden na str. 17. Pokud není zkratka uvedena (např. čtyřznaková notace), tak se předpokládá režim fronty FIFO.

Nutno poznamenat, že literatura [3, str. 269] uvádí klasifikaci jako posloupnost 6 znaků (použita jiná písmena než zde). Přičemž mezi pozici y a z je přidána pozice **E**, která obsahuje číslo, udávající početnost zdroje požadavků. Lze využít symbol ∞ pro vyjádření nekonečného zdroje.

$$\boxed{\text{A|B|x|y|E|z}}$$

Vzhledem k tomu, že se nejedná o často rozšířenou notaci, tak nebude používána ani v této práci.

1.3 Analýza systémů hromadné obsluhy

Samotná teorie hromadné obsluhy není schopna poskytnout odpověď na otázku, jaká konfigurace systému je optimální. Posouzení optimality konfigurace vždy záleží na konkrétních okolnostech a konkrétních prioritách daného subjektu (např. zdali je snaha maximalizovat vytížení linek obsluhy nebo raději mít v systémů kratší čekací doby, za cenu další linky obsluhy). Teorie hromadné obsluhy však poskytuje nástroje pro zkoumání systémů hromadné obsluhy, kdy je možné provádět analýzu jednotlivých konfigurací a na základě analyzovaných charakteristik systému se lze rozhodovat o potřebné konfiguraci systému. [1, str. 20]

1.3.1 Řešení systémů hromadné obsluhy

Řešením systémů hromadné obsluhy získáme potřebné charakteristiky o fungování systému. Pro řešení SHO existují dva různé přístupy: [3, str. 271]

- analytické řešení,
- simulační řešení.

Každé z těchto řešení má své výhody i nevýhody.

Analytické řešení spočívá v matematické analýze zkoumaného systému hromadné obsluhy. Na základě teorie pravděpodobnosti a příbuzných oborů jsou sestaveny pro jednotlivé konfigurace systémů jejich matematické modely. Analytické řešení předpokládá stabilizaci systému a zkoumá jeho charakteristiky v čase limitně blížícím se nekonečnu. Výhodou je přesné teoretické řešení, které uživatel modelu může pro známé konfigurace získat pouhým dosazením do již známých vzorců. Nevýhodou je, že analyticky nelze řešit všechny konfigurace systémů hromadné obsluhy, protože buď je jejich matematický model velmi složitý anebo není znám vůbec. [2, str. 89]

Simulační řešení se oproti tomu snaží napodobit chování reálného systému nikoliv matematickým, ale simulačním (počítačovým) modelem. Experimentuje se s počítačovým modelem systému (který je představen především programovými strukturami a algoritmy v paměti počítače). Generují se příchody zákazníků do systému a následně se měří a statisticky vyhodnocují různé pozorované veličiny, jako při pozorování reálného systému. Záleží jen na propracovanosti počítačového modelu, ale v principu je možné vytvořit model pro jakoukoliv konfiguraci SHO.

Nemá smysl vytvářet model již existujícího systému pro získání jeho současných charakteristik (jako např. vytížení linky obsluhy), protože tyto charakteristiky lze získat mnohem jednodušeji a přesněji přímo z reálného systému. Smyslem THO je ovšem zkoumání takových SHO, které neexistují anebo u existujících SHO zkoumat chování s jinými než aktuálními vstupy (např. očekávaný nárůst zákazníků, výroby apod.). [2, str. 88]

1.3.2 Charakteristiky systémů hromadné obsluhy

Ať již řešíme SHO analyticky nebo simulačně, tak nás zajímá, jak se systém bude chovat za určitých podmínek. Abychom dokázali vyjádřit a popsat ono chování, sledují se v SHO různé charakteristiky jednotlivých prvků i modelu celkového. Charakteristiky popisující fungování daného systém je možné rozdělit do několika skupin: [3, str. 270]

1. *Časové charakteristiky týkající se zákazníků.*

   Zajímavé jsou především jednotlivé doby strávené zákazníky v jednotlivých částech SHO. Protože se tyto doby liší zákazník od zákazníka, tak nás zajímají především

jejich průměrné (střední) hodnoty. Konkrétně se jedná o **střední dobu strávenou ve frontě**, **střední dobu strávenou v obsluze** a **střední dobu strávenou v systému**.

2. *Charakteristiky týkající se počtu zákazníků.*

   Jako v předešlém bodě postačují pouze průměrné hodnoty za všechny zákazníky, kdy sledujeme především **střední délku fronty** a **střední počet zákazníků v systému**.

3. *Pravděpodobnostní charakteristiky.*

   Pravděpodobnosti udávající setrvání systému a prvků v jednotlivých stavech, kdy se především zkoumá:

   - pravděpodobnost, že obslužná linka pracuje/nepracuje (vytížení linky),
   - pravděpodobnost, že zákazník bude muset čekat ve frontě (nehledě na dobu čekání),
   - pravděpodobnost, že v systému/frontě je právě n zákazníků (kde n vyjadřuje konkrétní počet zákazníků),
   - pravděpodobnost, že zákazník nebude z důvodu plné kapacity systému obsloužen (týká se jen systémů s ohraničenou frontou).

Pro snažší vyjádření je vhodné využít zástupných proměnných, které ovšem nebývají v literatuře jednotné a každý autor uvádí různé označení. Označení, které bude nadále používáno v této práci, uvádí tabulka 2. Použité označení vychází částečně z [4, 5].

Tabulka 2: Označení charakteristik použité v této práci

Označení	**Význam**
$E[W_Q]$	střední doba strávená ve frontě
$E[W_S]$	střední doba strávená v obsluze
$E[W]$	střední doba strávená v systému
$E[N_Q]$	střední délka fronty
$E[N_S]$	střední počet zákazníků v obsluze
$E[N]$	střední počet zákazníků v celém systému
P_Q	pravděpodobnost čekání zákazníka ve frontě
P_S	pravděpodonobost obsloužení zákazníka bez čekání
P_Z	pravděpodonobost odmítnutí
p_n	pravděpodobnost, že v systému je n zákazníků

U zmíněných charakteristik je třeba upozornit na rozdílnosti podle způsobu pozorování systému. Při popisu systému vnějším nezávislým pozorovatelem mohou být obdrženy jiné hodnoty než hodnoty vnímané z pohledu příchozího zákazníka. Příkladem může být systém D|D|1. Zákazníci přichází do systému pravidelně v intervalu *30 s*. Doba obsluhy zákazníka na lince je *15 s*. Pro vstupujícího zákazníka se jeví pravděpodobnost, že při vstupu bude systém prázdný rovna jedné ($p_0 = 1$). Vnější nezávislý pozorovatel ovšem „vidí“, že 15 s je systém prázdný a dalších 15 s je v systému jeden zákazník (tedy $p_0 = 0,5$ a $p_1 = 0,5$).

Výhodu mají systémy s Poissonovým vstupním tokem, protože tyto dva pohledy není nutné rozšlišovat. Podíl zákazníků, kteří při svém vstupu naleznou systém ve stavu N je stejný, jako podíl času v němž se systém ve stavu N nachází. Tato vlastnost se označuje výrazem „PASTA“ *(Poisson Arrivals See Time Averages)*. [6, str. 211]

1.3.3 Optimalizace systémů hromadné obsluhy

Získání potřebných charakteristik dané konfigurace systému bývá východiskem pro posouzení kvality a celkové výhodnosti dané konfigurace. Konkrétní způsob hledání optimální konfigurace se může lišit případ od případu, dle řešeného SHO a okolních požadavků a podmínek.

Pro optimalizaci lze využít nákladového ocenění jednotlivých prvků či charakteristik SHO (např. náklady na provoz každé linky obsluhy, ztráta příjmu spojená s odmítnutím zákazníka apod.).

2 ANALYTICKÉ ŘEŠENÍ

Analytické metody jsou založeny na analýze systému a následném sestavení matematického modelu s využitím dostupného matematického aparátu. Využívá se poznatků z matematických oborů jako je především teorie pravděpodobnosti a teorie náhodných procesů. Pokud pro daný SHO existuje ekvivalentní matematický model, stačí uživateli doplnit do vzorců vstupní parametry modelu a lze vypočítat přesné teoretické charakteristiky systému. Bohužel matematický model neexistuje pro jakýkoliv SHO, ale pouze pro jejich základní jednodušší příklady. Pro složitější SHO je matematický model buďto příliš složitý anebo není znám vůbec. Tyto SHO lze řešit pomocí simulace. [3, str. 272]

Z důvodu intenzivního využití matematického aparátu a především teorie pravděpodobnosti a teorie náhodných procesů je v následujících kapitolách uveden stručný úvod do teorie náhodných procesů, na základě které jsou odvozovány pro některé typy SHO jejich vzorce pro výpočet charakteristik systému. V následující části se předpokládá čtenářova základní znalost teorie pravděpodobnosti, se kterou se může seznámit např. v [9, 10].

2.1 Náhodné procesy

Matematické modely SHO jsou založeny na náhodných procesech, které mají specifické vlastnosti. Obecně lze náhodným procesem předpokládat jakýkoliv děj, který charakterizuje skutečnost, že se děje náhodně.

Definice 1. *Náhodný (stochastický) proces je systém* $\{X_t\}_{t\in T}$*, kde* X_t *jsou náhodné veličiny a* T *je množina parametrů. [1, str. 6]*

Obecně se náhodným procesem $\{X_t\}_{t\in T}$, který lze zapisovat i ve tvaru $\{X(t), t \in T\}$, rozumí posloupnost náhodných veličin, kterých nabývá při různých parametrech T, přičemž parametr T bude nadále chápán jako časová množina (nemusí tomu tak vždy být). Náhodným procesem tedy budeme rozumět posloupnost náhodných veličin X_t, kterých nabývá proces v čase t.

Obor hodnot náhodného procesu, kterých může náhodná veličina X_t nabývat, se nazývá stavovým prostorem procesu a budeme ho značit jako množinu S. Je zřejmé, že platí $X_t \in S$. Náhodná veličina musí nabývat stejné a neměnné věcné podstaty. Pokud má v čase t_1 náhodná veličina X_{t_1} význam např. počtu vozidel, která projela křižovatkou, tak

musí i náhodná veličina X_{t_2} v čase t_2 mít stejný význam, tj. počtu vozidel, která projela křižovatkou a nemůže se jednat např. o počet zákazníků na poště. [1, str. 6]

2.1.1 Markovovy procesy

Pro analytické řešení základních typů SHO, které využívají exponenciální rozložení pravděpodobnosti mezi příchody zákazníků a dob obsluhy (např. M|M|1), je důležitá znalost Markovových procesů. Jedná se o náhodné procesy, kdy následující stav procesu je závislý pouze na stavu předchozím a není podstatné, jakými stavy proces prošel v minulosti. Výše zmíněná skutečnost se označuje jako *Markovova závislost* anebo také jako paměť prvního řádu. [1, str. 11]

Markovovy procesy lze rozdělit do dvou skupin, dle způsobu vyjádření časové množiny. Rozlišujeme Markovovy procesy:

- s diskrétním časem (řetězce),
- se spojitým časem.

Markovovy řetězce s diskrétním časem

Proces probíhající v diskrétním čase může změnit svůj stav pouze v určitých okamžicích. Příkladem může být např. hod kostkou, kdy nás zajímá vždy stav po určitém počtu hodů, kde počtem hodů je přirozené číslo.

Definice 2. *Nechť $T = \{0, 1, 2, \dots\}$ a $\{X(t), t \in T\}$ je náhodný proces definovaný na spočetné množině stavů S. Proces se nazývá Markovův řetězec s diskétním časem, pokud pro $\forall s \in T$ platí: [4, str. 9]*

$$\mathcal{P}(X(t+s) = x' | X(t) = x, X(u) = x(u), u = 0, 1, \dots, t-1) = \mathcal{P}(X(t+s = x' | X(t) = x) \tag{3}$$

Podmínka (3) se nazývá *markovskou vlastností*. Zaručuje, že pravděpodobnost určitého stavu je závislá jen a pouze na bezprostředně předcházejícím stavu a není závislá na ostatních předcházejících stavech, kterými proces prošel.

Pro označení pravděpodobnosti, že se proces $\{X(t)\}$ nachází v okamžiku t (kde $t = 0, 1, 2, \dots$), v určitém stavu n (kde $n = 1, 2, \dots, s$), použijeme $p_n(t)$.

$$\mathcal{P}(X(t) = n) = p_n(t) \tag{4}$$

Pokud budou pravděpodobnosti stavů jednoho časového okamžiku sdruženy do vektoru, vzniká pravděpodobnostní vektor stavu systému v okamžiku t. Tento vektor budeme značit $\boldsymbol{p}(t)$: [2, str. 77]

$$\boldsymbol{p}(t) = [p_1(t), p_2(t), \ldots, p_s(t)] \tag{5}$$

Pro složky vektoru platí [2, str. 77]

$$0 \leq p_n(t) \leq 1 \qquad \text{pro } \forall t \in T, \forall n \in S \tag{6}$$

$$\sum_{n=1}^{s} p_n(t) = 1 \qquad \text{pro } \forall t \in T \tag{7}$$

Uvažujeme, že proces může v různém okamžiku t přecházet mezi svými stavy i a j $(i, j = 1, 2, \ldots, s)$. Pravděpodobnost, že přejde mezi těmito stavy v okamžiku t označme jako $p_{ij}(t)$. Jestliže pravděpodobnosti předchodu nejsou závislé na časovém okamžiku t, tzn. platí

$$\mathcal{P}(X(t+1) = j | X(t) = i) = p_{ij}(0) = p_{ij}(1) = p_{ij} \text{ pro } \forall t \in T \tag{8}$$

tak o Markovově řetězci hovoříme jako o homogenním v čase. [4, str. 78] Následují úvahy se týkají především těchto markovských homogenních řetězců.

Při předpokladu, že známe pravděpodobnostní vektor stavu systému v aktuálním okamžiku t označený jako $\boldsymbol{p}(t)$ je možné za použití pravděpodobností přechodů určit pravděpodobnosti jednotlivých stavů systému v bezprostředně následujícím okamžiku $t + 1$ pomocí vztahu: [4, str. 78]

$$p_j(t+1) = \sum_{n=1}^{s} p_n(t) \cdot p_{ij} \text{ pro } j = 1, 2, \ldots, s \tag{9}$$

Při využití vektorového zápisu pravděpodobností stavů a zapsáním pravděpodobnosti přechodů systému do tvaru matice

$$\boldsymbol{P} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1s} \\ p_{11} & p_{12} & \cdots & p_{1s} \\ \vdots & \vdots & \ddots & \vdots \\ p_{s1} & p_{s2} & \cdots & p_{ss} \end{bmatrix} \tag{10}$$

lze rovnici (9) pro $j = 1, 2, \ldots, s$, zapsat v maticovém tvaru následovně:

$$\boldsymbol{p}(t+1) = \boldsymbol{p}(t) \cdot \boldsymbol{P} \tag{11}$$

Matici $\boldsymbol{P}$ nazýváme maticí přechodů systému. Pro jednotlivé prvky platí: [2, str. 78]

$$0 \leq p_{ij} \leq 1 \qquad i, j = 1, 2, \ldots, s \tag{12}$$

$$\sum_{j=1}^{s} p_{ij} = 1 \qquad i = 1, 2, \ldots .s \tag{13}$$

Z rovnice (11) lze postupně pro jednotlivá $t = 0, 1, 2, \ldots$ dostat

$$\begin{aligned} \boldsymbol{p}(1) &= \boldsymbol{p}(0) \cdot \boldsymbol{P} \\ \boldsymbol{p}(2) &= \boldsymbol{p}(1) \cdot \boldsymbol{P} = \boldsymbol{p}(0) \cdot \boldsymbol{P} \cdot \boldsymbol{P} = \boldsymbol{p}(0) \cdot \boldsymbol{P} = \boldsymbol{p}(0) \cdot \boldsymbol{P}^2 \\ &\vdots \\ \boldsymbol{p}(t) &= \boldsymbol{p}(0) \cdot \boldsymbol{P}^t \end{aligned} \tag{14}$$

Matice $\boldsymbol{P}^t$ označuje matici přechodu systému po t krocích. Tato matice obsahuje jednotlivé prvky $p_{ij}^{(t)}$, význam takového prvku je pravděpodobnost přechodu ze stavu i do stavu j po t krocích.

Při zkoumání Markovovských řetězců nás častěji než zkoumání chování po určitém počtu kroků zajímají limitní vlastnosti řetězců. Tedy jejich chování pro čas limitně blížící se nekonečnu. Z pohledu zkoumání limitního chování řetězce, lze rozhodnout, zdali je řetězec: [4, str. 13]

- rozložitelný,
- periodický,
- ergodický.

Nerozložitelný řetězec splňuje podmínku, že pro $\forall i, j \in S, i \neq j, \exists t \in T$ takové, že $p_{ij}^{(t)} > 0$, jinými slovy, že po konečném počtu kroků se systém může dostat z jakéhokoliv stavu i do jakéhokoliv stavu j, tzn. stavy jsou dosažitelné. Pokud řetězec je rozložitelný, jeho matici přechodu systému $\boldsymbol{P}$ lze vhodným přečíslováním stavů upravit na tvar: [2, str. 79]

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D} \end{bmatrix} \tag{15}$$

kde $\boldsymbol{A}$ a $\boldsymbol{D}$ jsou čtvercové submatice matice $\boldsymbol{P}$ a $\boldsymbol{0}$ značí nulové matice.

Periodický řetězec osciluje mezi dvěma podmnožinami svých stavů. Aby toto bylo splněno, musí jít matici $\boldsymbol{P}$ vhodným přečíslováním stavů převést na tvar: [2, str. 79]

$$\begin{bmatrix} \boldsymbol{0} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{0} \end{bmatrix} \tag{16}$$

Liché mocniny matice $\boldsymbol{P}$ budou ve tvaru (16), zatimco sudé mocniny $\boldsymbol{P}$ budou ve tvaru (15). [2, str. 79] Periodický řetězec má tu vlastnost, že $\forall i \in S$ je $\delta(i) = NSD(t > 1 : p_{ij}^{(t)} > 0) > 1$ — stav má perodu $\delta(i)$). Opakem je aperiodický řetězec, pro který platí $\exists i \in S$ takové, že $NSD(t > 1 : p_{ii}^{(t)} > 0) = 1$ — stav nemá periodu.[3] [4, str. 13]

Ergodický řetězec je takový, že jeho matice $\boldsymbol{P}$ není rozložitelná ani periodická, jiným slovy je nerozložitelný a aperiodický. Pro ergodický řetězec existuje takový vektor rozdělení řetězce $\boldsymbol{p} = (p_n, n \in S)$, že platí

$$\boldsymbol{p} = \lim_{t\to\infty} p(t) = [p_1, p_2, \ldots, p_s] \tag{17}$$

Vektor $\boldsymbol{p}$ bývá nazýván vektorem stacionárního stavu systému. [2, str. 79]

Definice 3. *Markovův řetězec s maticí $\boldsymbol{P}$ má stacionární rozdělení $\boldsymbol{p}$, pokud existuje takový vektor $\boldsymbol{p}$, že: [4, str. 14]*

$$\boldsymbol{p} = \boldsymbol{p} \cdot \boldsymbol{P} \tag{18}$$

$$1 = 1 \cdot \boldsymbol{p} \tag{19}$$

Na základě rovnice (18) lze stanovit za podmínky (19) jednotlivé složky vektoru $\boldsymbol{p}$. Přičemž podmínku (19) lze vyjádřit pomocí jednotlivých složek p_n následovně:

$$\sum_{n=1}^{s} p_n = 1 \tag{20}$$

Markovovy procesy se spojitým časem

Proces se spojitým časem může změnit svůj stav v libovolných časových okamžicích. Příkladem je model SHO, kdy příchody i odchody zákazníků probíhají spojitě v čase.

Definice 4. *Nechť $T = \langle 0, \infty)$ a $\{X(t), t \in T\}$ je náhodný proces definovaný na spočetné množině stavů S. Proces se nazývá Markovův proces se spojitým časem, pokud pro $\forall s \in T$ platí: [4, str. 10]*

$$\mathcal{P}(X(t+s) = x' | X(t) = x, X(u) = x(u), 0 \leq u < t) = \mathcal{P}(X(t+s = x' | X(t) = x) \tag{21}$$

kde

$$\mathcal{P}(X(t) = x, X(u) = x(u), 0 \leq u \leq t) > 0 \tag{22}$$

[3] $NSD(n_1, n_2, \ldots)$ označuje největší společný dělitel čísel $n_1, n_2, \ldots$

Pokud navíc platí pro $\forall t \in T$

$$\mathcal{P}(X(t+s) = j | X(t) = x)) = p_{ij}(s) \tag{23}$$

označuje se proces za *časově homogenní*.

Předmětem zkoumání jsou místo samotných pravděpodobností jejich jisté limitní veličiny. Uvažujme limitní veličinu označenou jako a_{ij}

$$a_{ij} = \begin{cases} \lim\limits_{\Delta t \to \infty} \dfrac{p_{ij}(t+\Delta t)}{\Delta t} & \text{pro } i \neq j \\ \lim\limits_{\Delta t \to \infty} \dfrac{1 - p_{ij}(t+\Delta t)}{\Delta t} & \text{pro } i = j \end{cases} \tag{24}$$

Limity uvedené ve vzahu (24) se nazývají jako *intenzity pravděpodobností přechodů*. [2, str. 83] Zatímco pro $i \neq j$ mají intenzity a_{ij} význam intenzity pravděpodobnosti přechodu mezi stavy i a j (analogicky k pravděpodobnosti p_{ij}), tak pro $i = j$ vyjadřuje a_{ii} intenzitu pravděpodobnosti opuštění stavu i (narozdíl od pravděpodobnost p_{ii} setrvání ve stavu i). Pomocí intenzit pravděpodobnosti přechodu lze vyjádřit matici pravděpodobnosti přechodu mezi okamžiky t a $t + \Delta t$ následovně: [2, str. 83]

$$\boldsymbol{P}(t, t+\Delta t) \cong \begin{bmatrix} 1 - a_{11}\Delta t & a_{12}\Delta t & \cdots & a_{1s}\Delta t \\ a_{21}\Delta t & 1 - a_{22}\Delta t & \cdots & a_{2s}\Delta t \\ \vdots & \vdots & \ddots & \vdots \\ a_{s1}\Delta t & a_{s2}\Delta t & \cdots & 1 - a_{ss}\Delta t \end{bmatrix} = \boldsymbol{E} + \Delta t \begin{bmatrix} -a_{11} & a_{12} & \cdots & a_{1s} \\ a_{21} & -a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ a_{s1} & a_{s2} & \cdots & -a_{ss} \end{bmatrix} \tag{25}$$

kde $\boldsymbol{E}$ je jednotková matice. Protože pro pravděpodosti p_{ij} platí vztah (13), tak musí platit, že řádkové součy matice intenzit a_{ij} jsou nulové, tedy: [2, str. 83]

$$-a_{ii} + \sum_{\substack{j=1 \\ j \neq i}}^{s} a_{ij} = 0, \text{pro } \forall i \in S \tag{26}$$

Při označení matice intenzit pravděpodobností přechodu ve vztahu (25) jako $\boldsymbol{Q}$, lze zapsat vztah pro absolutní pravděpodobnosti mezi okamžiky t a $t + \Delta t$ následovně: [2, str. 83]

$$\boldsymbol{p}(t+\Delta t) = \boldsymbol{p}(t) \cdot \boldsymbol{P}(t+\Delta t) = \boldsymbol{p}(t) \cdot (\boldsymbol{E} + \Delta t \boldsymbol{Q}) \tag{27}$$

Postupnou úpravou rovnice (27) dostaneme

$$\frac{\boldsymbol{p}(t+\Delta t) - \boldsymbol{p}(t)}{\Delta t} = \boldsymbol{p}(t) \cdot \boldsymbol{Q} \tag{28}$$

dále limitním přechodem pro $\Delta t \to 0$ a dle definice derivace

$$\lim_{\Delta t \to 0} \frac{\boldsymbol{p}(t+\Delta t) - \boldsymbol{p}(t)}{\Delta t} = \boldsymbol{p}'(t) \tag{29}$$

čímž získáme vektorovou diferenciální rovnici pro vektor absolutních pravděpodobností: [2, str. 84]

$$\boldsymbol{p}'(t) = \boldsymbol{p}(t) \cdot \boldsymbol{Q} \tag{30}$$

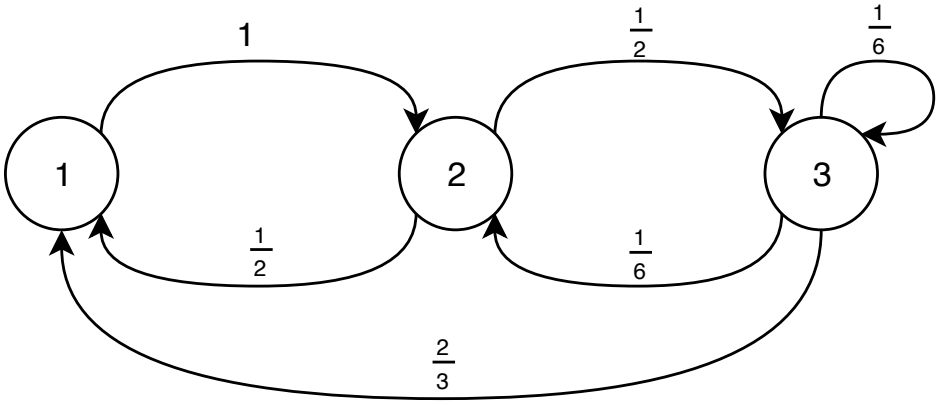
Protože v matematických modelech SHO nás nezajímají přechodové jevy procesu, ale především limitní charakteristiky stacionárního stavu, lze předpokládat, že pro $t \to \infty$ platí podmínka $\boldsymbol{p}'(t) = 0$. Dosazením do (30) získáme: [2, str. 84]

$$\boldsymbol{p}(t) \cdot \boldsymbol{Q} = 0 \tag{31}$$

2.1.2 Grafová reprezentace

Zajímavá a užitečná je reprezentace pravděpodobností přechodů nebo intenzit přechodů pomocí grafického vyjádření orientovaným grafem. Používá se především v případě, kdy matice pravděpodobností přechodů (intenzit) je řidká a obsahuje mnoho nulových prvků. [4, str. 17]

Definice 5. *Nechť P je matice pravděpodobností přechodu Markovova řetězce s diskrétním časem a množinou stavů S. Poté rozumíme přechodovým grafem řetězce ohodnocený graf $G = (V, H, c)$, kde $V = S$ je množina vrcholů, $H = \{(i,j) \in V \times V : p_{ij} > 0\}$ je množina orientovaných hran a $c : H \to (0, 1\rangle$ je ohodnocení orientovaných hran určené $c((i,j)) = p_{ij}$. [4, str. 33]*

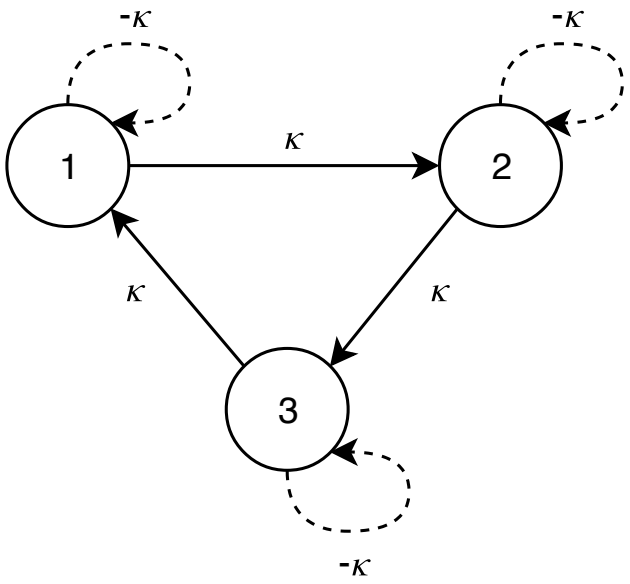


Obrázek 5: Grafické znázornění pravděpodobností přechodu pomocí grafu. Zpracování autora dle [4, str. 18]

Pro konkrétní ukázku předpokládejme, že máme pravděpodobností přechodů mezi stavy řetězce definovány v následující matici:

$$\boldsymbol{P} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{2}{3} & \frac{1}{6} & \frac{1}{6} \end{bmatrix} \tag{32}$$

Příslušné grafové znázornění pravděpodobnosti přechodů je znázorněno na obrázku 5. V případě procesu se spojitým časem, se pro ohodnocení hran může také využít $p_{ij}(\Delta t)$. Ohodnocení je pak tedy určené $c((i,j)) = p_{ij}(\Delta t)$. [4, str. 19]



Obrázek 6: Grafické znázornění intenzity pravděpodobností přechodu pomocí grafu. Zpracování autora.

Kromě pravděpodobností přechodů mezi jednotlivými stavy, lze podobně vyjádřit intenzity pravděpodobností přechodů. Obrázek 6 znázorňuje reprezentaci následující matice intenzit pravděpodobností:

$$\boldsymbol{Q} = \begin{bmatrix} -\kappa & \kappa & 0 \\ 0 & -\kappa & \kappa \\ \kappa & 0 & -\kappa \end{bmatrix} \tag{33}$$

Pro zjednodušení je možné z grafu na obrázku 6 vynechat zpětné smyčky *(zakresleny přerušovanou čárou)*, které se dají snadno vždy dopočítat podle ostatních hran, tak aby byla splněna pro řádek matice podmínka daná rovnicí (26).

2.1.3 Poissonův náhodný proces

Předpokládejme Markovovův proces se spojitým časem. Jednotlivé stavy tohoto náhodného procesu představují určitý počet událostí, které v daném čase nastaly. Předpokládá se pouze možný vznik událostí, nikoliv jejich zánik. Ve velmi krátkém časovém okamžiku Δt je tedy možné předpokládat následující: [2, str. 84]

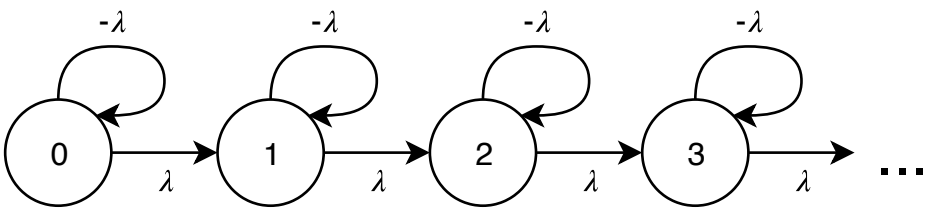
- nastane právě jedna událost,
- nenastane žádná událost.

Náhodný proces tedy může buďto setrvávat v současném stavu (událost nenastala) anebo nastane přechod ze současného do následujícího „sousedního“ stavu (nastala právě jedna událost). Pravděpodobnost přechodu a tedy i intenzita pravděpodobnosti přechodu je konstantní a nezávislá na aktuálním stavu. Pokud označíme intenzitu pravděpodobnosti přechodu do následujícího stavu jako λ, platí:: [2, str. 84]

$$a_{ij} = \begin{cases} -\lambda & \text{pro } j = i \\ \lambda & \text{pro } j = i + 1 \\ 0 & \text{pro } j \notin \{i, i+1\} \end{cases} \tag{34}$$

Dle předpisu (34) lze sestavit matici intenzity pravděpodobnosti přechodů $\boldsymbol{Q}$:

$$\boldsymbol{Q} = \begin{bmatrix} -\lambda & \lambda & 0 & 0 & \cdots \\ 0 & -\lambda & \lambda & 0 & \cdots \\ 0 & 0 & -\lambda & \lambda & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{35}$$



Obrázek 7: Přechodový graf Poissonova náhodného procesu. Zpracování autora dle [4, str. 18]

Vyjádření přechodovým grafem tohoto procesu znázorňuje obrázek 7. Pravděpodobnosti jednotlivých vztahů lze získat z rovnice (30), čímž dostaneme soustavu diferenciál-

ních rovnic

$$\begin{aligned} p_0'(t) &= && -\lambda p_0(t) \\ p_1'(t) &= \lambda p_0(t) && -\lambda p_1(t) \\ p_2'(t) &= \lambda p_1(t) && -\lambda p_2(t) \\ &\vdots \end{aligned} \tag{36}$$

Je zřejmé, že pro $n \geq 1$ platí rekurentní vztah

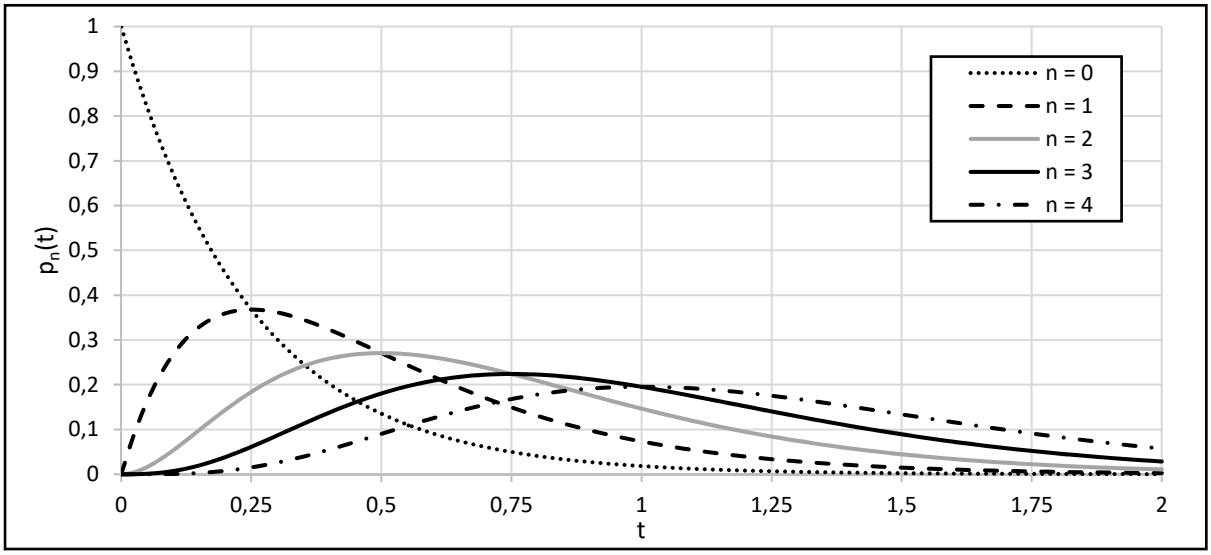
$$p_n'(t) = \lambda p_{n-1}(t) - \lambda p_n(t) \tag{37}$$

Získat výjadření jednotlivých pravděpodobností $p_n(t)$ lze řešením soustavy diferenciálních rovnic (36) za počáteční podmínky $p_0(t) = 1$ (tedy v čase $t = 0$ stav procesu bude $n = 0$, tzn. nevyskytla se doposud žádná událost)

$$p_n(t) = \frac{(t\lambda)^n}{n!} e^{-\lambda t} \tag{38}$$

Obdržený vzorec (38) vyjadřuje Poissonovo rozdělení pravděpodobnosti s parametrem λ, který udává střední počet vyskytujících se událostí za jednotku času. [2, str. 84]

Obrázek 8 zobrazuje graf průběhu pravděpodobností jednotlivých stavů v závislosti na čase t, kde $t \in \langle 0, 2 \rangle$. Použitý parametr procesu $\lambda = 4$.



Obrázek 8: Pravděpodobnosti stavů Poissonova procesu s parametrem $\lambda = 4$. Zpracování autora.

Poissonovo rozdělení pravděpodobnosti vyjadřuje pravděpodobnost výskytu n událostí za určitý časový interval $\langle 0, t \rangle$. Často je ovšem potřebné znát rozdělení pravděpodobnosti časových intervalů mezi jednotlivými událostmi. To lze odvodit následující jednoduchou

úvahou — pravděpodobnost, že v daném intervalu $\langle 0, t\rangle$ nedošlo k žádné události ($n = 0$) lze získat dosazením do (38): [2, str. 84]

$$p_0(t) = \frac{(t\lambda)^0}{0!} e^{-\lambda t} = e^{-\lambda t} \tag{39}$$

Pravděpodobnost, že v intervalu $\langle 0, t\rangle$ došlo alespoň k jedné události je doplňkovým jevem k (39), platí tedy

$$\overline{p_0(t)} = 1 - p_0(t) = 1 - e^{-\lambda t} \tag{40}$$

Přičemž pravděpodobnost uvedenou v (40) lze také interpretovat tak, že se jedná o pravděpodobnost, že náhodná proměnná doby prvního vstupu τ je menší než t, tedy: [2, str. 84]

$$\mathcal{P}(\tau \leq t) = \overline{p_0(t)} = 1 - e^{-\lambda t} \tag{41}$$

Vzorec (41) není nic jiného než distribuční funkce doby mezi dvěma událostmi. Odpovídající funkci hustoty pravděpodobnosti doby mezi dvěma událostmi lze získat derivací distribuční funkce (41), čímž dostáváme *exponenciální rozdělení pravděpodobnosti*. Je třeba dodat podmínku, že čas t nemůže být záporný ($t \geq 0$) a stejně tak u spojitého rozdělení pravděpodobnosti platí, že pravděpodobnost $\mathcal{P}(X = c) = 0$, kde $c \in \mathbb{R}$. Po aplikování předešlého získáváme funkci hustoty pravděpodobnosti: [2, str. 85]

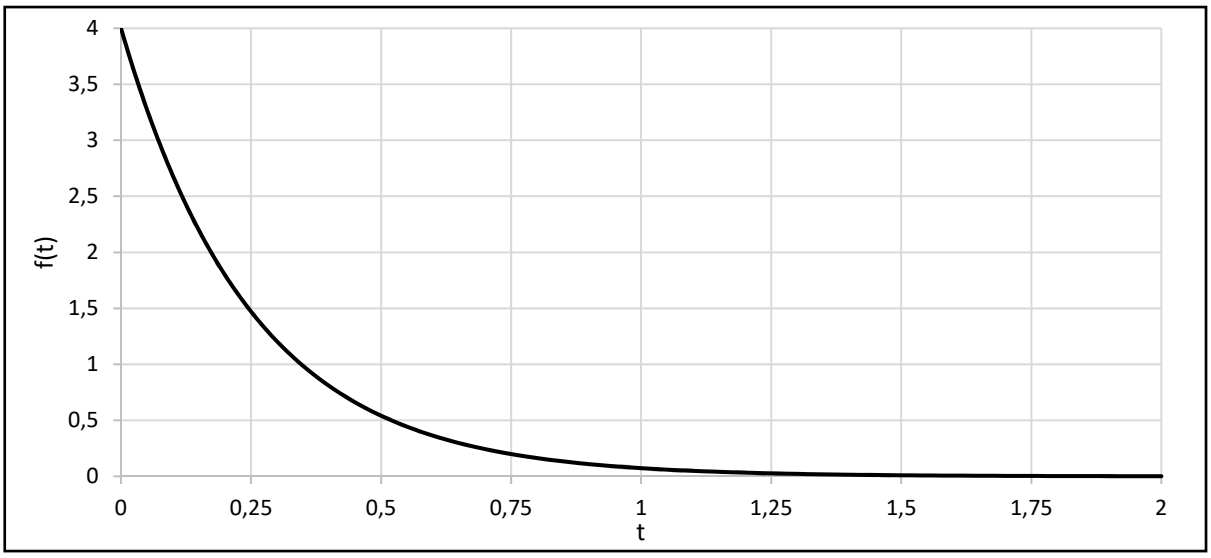
$$f(t) = \begin{cases} \lambda e^{-\lambda t} & \text{pro } t > 0 \\ 0 & \text{pro } t \leq 0 \end{cases} \tag{42}$$

Znázornění průběhu funkce hustoty exponenciálního rozdělení pravděpodobnosti s použitým parametrem $\lambda = 4$ je znázorněno na obr. 9 . Střední hodnota náhodné veličiny X, která má exponenciálního rozdělení s parametrem λ ($X \sim Exp(\lambda)$) je rovna převrácené hodnotě parametru λ

$$E[X] = \frac{1}{\lambda} \tag{43}$$

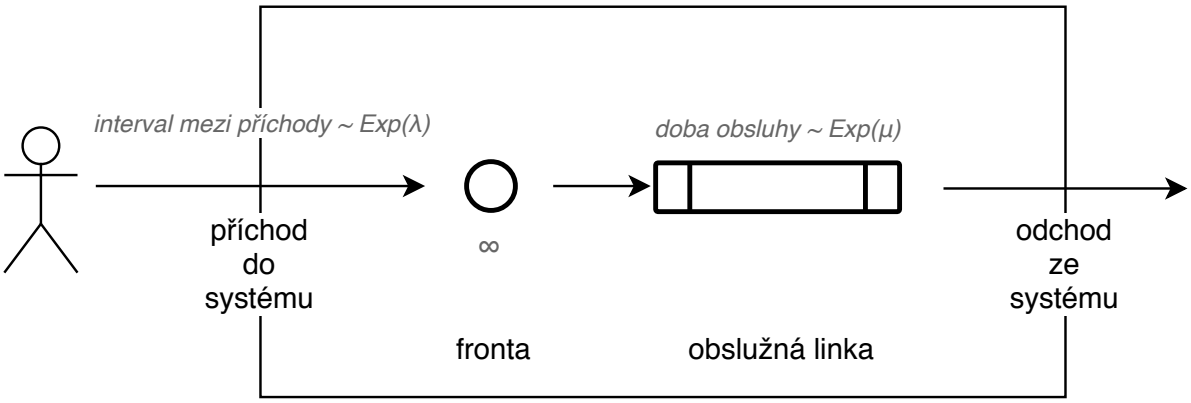
2.2 Systém M|M|1

Systém M|M|1 je znám jako jeden z nejjednodušších modelů SHO. Jeho využití nebývá tak časté, ale díky své jednoduchosti na něm lze demonstrovat odvození základních vzorců pro výpočet charakteristik systému. V systému M|M|1 tvoří příchod zákazníků Poissonův proces s parametrem λ. Model obsahuje jednu frontu, která není ohraničena. Obsluha



Obrázek 9: Funkce hustoty pravděpodobnosti exponenciálního rozdělení s parametrem $\lambda = 4$. Zpracování autora.

zákazníků probíhá pouze na jediné lince, kde doba obsluhy má exponenciální rozdělení s parametrem μ. [2, str. 91] Schématické znázornění SHO je ilustrováno na obr. 10.



Obrázek 10: Schématické znázornění systému M|M|1. Zpracování autora.

Systém hromadné obsluhy považujeme za Markovský proces $\{N(t), t \geq 0\}$, který nabývá různých stavů ze stavové množiny $S = \{0, 1, 2, \dots\}$. Jednotlivé stavy značí počet zákazníků v systému. Např. stav $N = 3$ znamená, že v systému jsou celkem 3 zákazníci, z toho jeden je právě obsluhován a zbylí dva čekají ve frontě.

Předpokládejme o procesu $\{N(t)\}$, že je složen ze dvou procesů. Náhodný proces $\{A(t)\}$, který modeluje příchod zákazníků do systému a náhodný proces $\{B(t)\}$, který modeluje naopak jejich odchod. Počet zákazníku v systému, lze tedy získat ze vztahu: [4,

str. 31]

$$N(t) = A(t) - B(t) \tag{44}$$

Uvažujme dostatečně malý časový interval Δt, pro který budeme počítat pravděpodobnost příchodu, popř. dokončení obsluhy zákazníka. Pravděpodobnosti příchodu zákazníka jsou dány charakteristikou Poissonova procesu a lze je získat ze vzorce (38): [4, str. 33]

$$\mathcal{P}(A(\Delta t) = n) = \begin{cases} e^{-\lambda\Delta t} = 1 - \lambda\Delta t + o(\Delta t) & \text{pro } n = 0 \\ \Delta t\lambda e^{-\lambda\Delta t} = \lambda\Delta t + o(\Delta t) & \text{pro } n = 1 \\ \dfrac{(\Delta t\lambda)^n}{n!} e^{-\lambda\Delta t} = o(\Delta t) & \text{pro } n \geq 2 \end{cases} \tag{45}$$

Pravděpodobnost, že v intervalu Δt bude právě ukončena obsluha zákazníka na lince obsluhy je totožná s pravděpodobností, že zbytková doba obsluhy $\tau \leq \Delta t$. Tuto pravděpodobnost lze vyjádřit z (41). Zbytková doba obsluhy τ má stejné rozložení pravděpodobnosti jako celková doba obsluhy ($\tau \sim Exp(\mu)$). [4, str. 33]

$$\mathcal{P}(B(\Delta t) = 1) = \mathcal{P}(\tau \leq \Delta t) = 1 - e^{\mu\Delta t} = \mu\Delta t + o(\Delta t) \tag{46}$$

Symbol $o(\Delta t)$ ve vzorci (45) představuje tzv. zbytkovou funkci pravděpodobnosti. Pokud ovšem volíme velmi malý časový interval Δt, tak hodnoty funkce $o(\Delta t)$ jsou v porovnání s celkovou pravděpodobností natolik malé, že je lze zanedbat, což zjednodušší další výpočty. Nadále tedy předpokládáme aproximaci: [1, str. 27]

$$o(\Delta t) \cong 0 \tag{47}$$

Následně se budeme zabývat jednotlivými pravděpodobnostmi přechodu mezi různými stavy systému v uvažovaném časovém intervalu Δt. Pravděpodobnost přechodu ze stavu 0 do stavu 1 odpovídá pravděpodobnosti příchodu zákazníka do systému. [4, str. 33] Opačným jevem je setrvání ve stavu 0. Pravděpodobnost lze získat dosazením do vzorce (45):

$$p_{00}(\Delta t) = \mathcal{P}(A(\Delta t) = 0) = 1 - \lambda\Delta t \tag{48}$$

$$p_{01}(\Delta t) = \mathcal{P}(A(\Delta t) = 1) = \lambda\Delta t \tag{49}$$

Je-li systém v určitém stavu i, pak to znamená, že jeden zákazník je v obsluze a zbylých $i - 1$ čeká ve frontě. Pro pravděpodobnost přechodu do stavu $i + 1$ platí, že je totožná s pravděpodobností jevu, kdy v čase Δt přijde do systému zákazník, ale zároveň žádný

neodejde. Podobně pravděpodobnost pro přechod ze stavu i do stavu $i-1$ je stejná jako pravděpodobnost jevu, že skončí obsluha zákazníka, ale zároveň žádný nepřijde. Pravděpodobnost setrvání ve stavu i je stejná pravděpodobnosti, že nepřijde žádný zákazník a zároveň neskončí obsluha. [4, str. 34] Pravděpodobností jsou tedy odvozeny na základě (45) a (46):

$$p_{i,i-1}(\Delta t) = \mathcal{P}(A(\Delta t) = 0)\mathcal{P}(\tau \leq \Delta t) = (1 - \lambda\Delta t)(\mu\Delta t) \cong \mu\Delta t \tag{50}$$

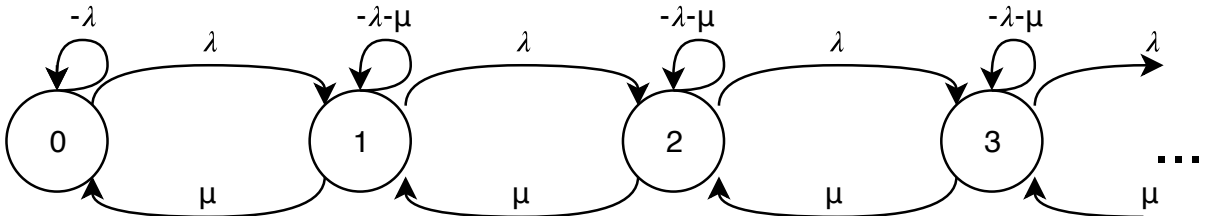
$$p_{i,i+1}(\Delta t) = \mathcal{P}(A(\Delta t) = 1)\mathcal{P}(\tau > \Delta t) = (\lambda\Delta t)(1 - \mu\Delta t) \cong \lambda\Delta t \tag{51}$$

$$p_{ii}(\Delta t) = \mathcal{P}(A(\Delta t) = 0)\mathcal{P}(\tau > \Delta t) = (1 - \lambda\Delta t)(1 - \mu\Delta t) \cong 1 - \lambda\Delta t - \mu\Delta t \tag{52}$$

Aplikováním limit (24) lze získat vyjádření pro prvky matice intenzity pravděpodobností q_{ij}: [4, str. 34]

$$q_{ij} = \begin{cases} -\lambda & \text{když } i, j = 0 \\ \lambda & \text{když } j = i + 1 \\ \mu & \text{když } j = i - 1 \\ -\lambda - \mu & \text{když } i = j, i \leq 1 \\ 0 & \text{když } |i - j| > 1 \end{cases} \tag{53}$$

Obrázek 11 znázorňuje ilustraci (53) přechodovým grafem intenzit pravděpodobností mezi stavy.



Obrázek 11: Přechodový graf intenzit pravděpodobností systému M|M|1. Zpracování autora dle [4, str. 28]

Analytické řešení určuje pravděpodobnosti pro stabilizovaný systém, tj. takový systém jehož pravděpodobnostní a další charakteristiky se pro $t \to \infty$ ustálí. Systém M|M|1 je modelován jako proces vzniku a zániku. Pro tyto procesy existuje nutná a postačující podmínka pro stabilizaci procesu: [4, str. 28]

$$\sum_{n=1}^{\infty} \frac{\lambda_0\lambda_1 \ldots \lambda_{n-1}}{\mu_1\mu_2 \ldots \mu_n} < \infty \tag{54}$$

Za této podmínky platí, že:

$$p_n = p_0 \frac{\lambda_0 \lambda_1 \dots \lambda_{n-1}}{\mu_1 \mu_2 \dots \mu_n} \text{ pro } n \geq 1 \tag{55}$$

kde

$$p_0 = \left(1 + \sum_{n=1}^{\infty} \frac{\lambda_0 \lambda_1 \dots \lambda_{n-1}}{\mu_1 \mu_2 \dots \mu_n}\right)^{-1} \tag{56}$$

Aby byla splněna podmínka stability systému (54), musí pro systém M|M|1 platit: [4, str. 34]

$$\rho = \frac{\lambda}{\mu} < 1 \tag{57}$$

Dosazením do vztahů (55) a (56) lze poté dostat tzv. Erlangovy vzorce:

$$p_n = (1 - \rho)\rho^n \text{ pro } n \geq 0 \tag{58}$$

Vzorec (58) vyjadřuje pravděpodobnost, že se v systému M|M|1 nachází právě n zákazníků. Pravděpodobnost čekání zákazníka ve frontě:

$$P_Q = 1 - p_0 = \rho \tag{59}$$

Pravděpodobnost obsloužení zákazníka bez čekání:

$$P_S = p_0 = 1 - \rho \tag{60}$$

Střední počet zákazníků v celém systému lze vyjádřit následovně: [5, str 60]

$$E[N] = \sum_{n=0}^{\infty} n p_n = \frac{\rho}{1 - \rho} = \frac{\lambda}{\mu - \lambda} \tag{61}$$

Protože existuje pouze 1 linka obsluhy, tak střední délku fronty lze vyjádřit:

$$E[N_Q] = \sum_{n=1}^{\infty} (n - 1) p_n = \frac{\rho^2}{1 - \rho} = \frac{\lambda^2}{\mu(\mu - \lambda)} \tag{62}$$

Střední počet zákazníků v obsluze (můžeme interpretovat také jako vytížení linky obsluhy) lze odvodit z výše uvedených charakteristik:

$$E[N_S] = E[N] - E[N_Q] = \rho = \frac{\lambda}{\mu} \tag{63}$$

Velmi užitečná je v THO tzv. Littleova formule, která vyjadřuje střední počet zákazníků v SHO v závislosti na intenzitě příchodů zákazníků a střední době strávené zákazníkem v systému. Je známa jako vzorec: [8, str. 1]

$$L = \lambda W \tag{64}$$

kde L je střední počet zákazníků v systému, λ je střední intenzita příchodů zákazníků do systému a W je střední doba strávená zákazníkem v systému. Formule není závislá na pravděpodobnostním rozložení příchodů zákazníků, ani na pravděpodobnostním rozložení doby obsluhy v systému. Systémem nemusí být nutně jen SHO jako celek, ale za „systém“ lze považovat např. pouze vstup zákazníků a samotnou frontu zákazníků. [8, str. 2]

Střední dobu strávenou zákazníkem v systému lze pomocí (64) vyjádřit jako:

$$E[W] = \frac{E[N]}{\lambda} = \frac{\rho}{\lambda(1-\rho)} = \frac{1}{\mu - \lambda} \tag{65}$$

Obdobně s využitím (64) lze vyjádřit střední dobu strávenou ve frontě:

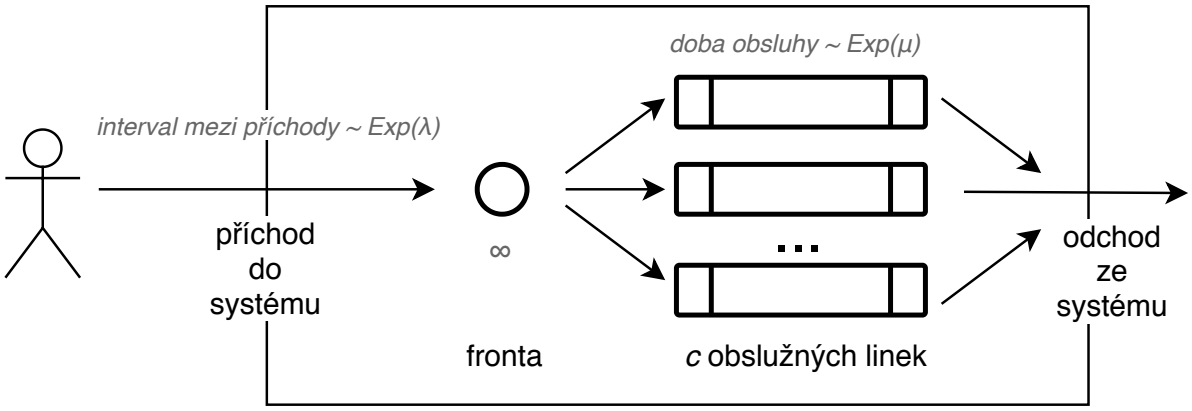
$$E[W_Q] = \frac{E[N_Q]}{\lambda} = \frac{\rho}{\mu(1-\rho)} = \frac{\rho}{\mu - \lambda} \tag{66}$$

Střední doba strávená zákazníkem v obsluze:

$$E[W_S] = E[W] - E[W_Q] = \frac{1}{\mu} \tag{67}$$

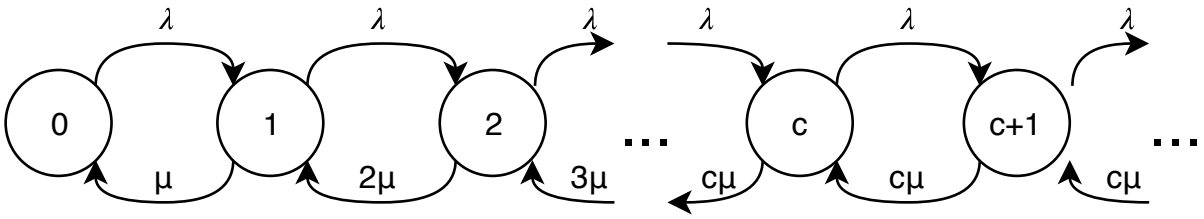
2.3 Systém M|M|c

V systému M|M|c stejně jako v M|M|1 tvoří příchody zákazníků Poissonův proces a doba obsluhy má exponenciální rozdělení pravděpodobnosti. Fronta nemá omezenou kapacitu. K obsluze zákazníka slouží více paralelně pracujících linek obsluhy, jejichž počet je charakterizován proměnnou c. Pro linky obsluhy platí, že jsou pro zákazníka navzájem rovnocenné a plně zastupitelné. Všechny linky obsluhy mají stejnou charakteristiku, tj. dobu obsluhy tvořenou exponenciálním rozdělením pravděpodobnosti se stejnou střední intenzitou μ. Systém je znázorněn na obr. 12.



Obrázek 12: Schématické znázornění systému M|M|c. Zpracování autora.

Na obrázku 13 jsou znázorněny intenzity pravděpodobnosti přechodů mezi jednotlivými stavy systému. Pokud je v systému méně zákazníků než je linek obsluhy, intezita pravděpodobnosti přechodu ze stavu n do $n-1$ je závislá na aktuálním počtu zákazníků — čím větší počet zákazníků, tím větší pravděpodobnost ukončení obsluhy některého ze zákazníků. Jakmile je v systému c a více zákazníků, tak je intezita pravděpodobnosti již pouze závislá na počtu linek obsluhy v systému, nehledě na aktuální počet zákazníků v systému (stav systému).



Obrázek 13: Přechodový graf intenzit pravděpodobností systému M|M|c. Zpracování autora dle [5, str. 67]

Analytické vzorce pro výpočet charakteristik systému M|M|c budou uvedeny bez jejich odvozování. Odvození případných vzorců lze nalézt např. v [1, str. 26], [2, str. 102] nebo [5, str. 67].

Zavedeme pomocné proměnné r a ρ:

$$r = \frac{\lambda}{\mu} \tag{68}$$

$$\rho = \frac{r}{c} = \frac{\lambda}{c\mu} \tag{69}$$

Nutnou a postačující podmínkou pro stabilizaci systému je: [1, str. 31]

$$\rho < 1 \tag{70}$$

Pravděpodobnost jednotlivých stavů systému: [5, str. 67]

$$p_n = \begin{cases} \dfrac{\lambda^n}{n!\mu^n} p_0 & \text{když } 0 \leq n < c \\ \dfrac{\lambda^n}{c^{n-c} c!\mu^n} p_0 & \text{když } n \geq c \end{cases} \tag{71}$$

$$p_0 = \left(\frac{r^c}{c!(1-\rho)} + \sum_{n=0}^{c-1} \frac{r^n}{n!} \right)^{-1} \tag{72}$$

Pravděpodobnost čekání zákazníka ve frontě: [5, str. 69]

$$P_Q = \frac{r^c p_0}{c!(1-\rho)} = p_c \frac{1}{1-\rho} \tag{73}$$

Pravděpodobnost obsluhy zákazníka bez čekání:

$$P_S = 1 - P_Q = 1 - \frac{r^c p_0}{c!(1-\rho)} \tag{74}$$

Střední počet zákazníků ve frontě: [5, str. 68]

$$E[N_Q] = \frac{r^c \rho}{c!(1-\rho)^2} p_0 \tag{75}$$

Střední počet zákazníků v obsluze:

$$E[N_S] = \frac{\lambda}{\mu} = r \tag{76}$$

Střední počet zákazníků v systému:

$$E[N] = E[N_S] + E[N_Q] = r + \frac{r^c \rho}{c!(1-\rho)^2} p_0 \tag{77}$$

Střední doba strávená zákazníkem ve frontě: [5, str. 69]

$$E[W_Q] = \frac{E[N_Q]}{\lambda} = \frac{r^c}{c!(c\mu)(1-\rho)^2} p_0 \tag{78}$$

Střední doba strávená zákazníkem v obsluze:

$$E[W_S] = \frac{1}{\mu} \tag{79}$$

Střední doba strávená zákazníkem v systému:

$$E[W] = E[W_S] + E[W_Q] = \frac{1}{\mu} + \frac{E[N_Q]}{\lambda} = \frac{r^c}{c!(c\mu)(1-\rho)^2} p_0 \tag{80}$$

2.4 Systém M|M|c|K

Systém M|M|c|K je oproti M|M|c omezen počtem zákazníků, kterých může být v systému maximálně K. V případě, že je v systému K zákazníků, jsou další zákazníci při příchodu odmítáni. Fronta v systému je ohraničená a její maxímální kapacita je $K - c$ zákazníků.

Protože systém odmítá zákazníky, neexistuje podmínka omezující stabilizaci systému. Systém je tedy narozdíl od předcházejících modelů stabilní i pro $\rho \geq 1$. Oproti předcházejícím modelů přibývá důležitá charakteristika, kterou je pravděpodobnost odmítnutí

zákazníka P_Z. Vzorce jsou opět uváděny bez odvození, které lze nalézt např. v [4, str. 51]. Předpokládá se zavedení pomocných proměnných r a ρ dle (68) a (69).

Pravděpodobnost jednotlivých stavů systému: [5, str. 76]

$$p_n = \begin{cases} \dfrac{\lambda^n}{n!\mu^n} p_0 & \text{když } 0 \leq n < c \\ \dfrac{\lambda^n}{c^{n-c} c!\mu^n} p_0 & \text{když } c \leq n \leq K \end{cases} \tag{81}$$

$$p_0 = \begin{cases} \left[\dfrac{r^c}{c!} \left(\dfrac{1-\rho^{K-c+1}}{1-\rho} \right) + \displaystyle\sum_{n=0}^{c-1} \dfrac{r^n}{n!} \right]^{-1} & \text{když } \rho \neq 1 \\ \left[\dfrac{r^c}{c!} (K-c+1) + \displaystyle\sum_{n=0}^{c-1} \dfrac{r^n}{n!} \right]^{-1} & \text{když } \rho = 1 \end{cases} \tag{82}$$

Pravděpodobnost čekání zákazníka ve frontě:

$$P_Q = \sum_{n=c}^{K-1} p_n \tag{83}$$

Pravděpodobnost obsluhy zákazníka bez čekání:

$$P_S = \sum_{n=0}^{c-1} p_n \tag{84}$$

Pravděpodobnost odmítnutí zákazníka:

$$P_Z = p_K \tag{85}$$

Střední počet zákazníků ve frontě: [5, str. 78]

$$E[N_Q] = \sum_{n=c+1}^{K} (n-c) p_n = \frac{p_0 r^c \rho}{c!(1-\rho)^2} \left[1 - \rho^{K-c+1} - (1-\rho)(K-c+1)\rho^{K-c} \right] \tag{86}$$

Střední počet zákazníků v obsluze: [4, str. 54]

$$E[N_S] = \frac{\lambda}{\mu}(1-P_Z) = r(1-P_Z) \tag{87}$$

Střední počet zákazníků v systému:

$$E[N] = E[N_S] + E[N_Q] \tag{88}$$

Střední doba strávená zákazníkem ve frontě: [4, str. 55]

$$E[W_Q] = \frac{E[N_Q]}{\lambda(1-P_Z)} \tag{89}$$

Střední doba strávená zákazníkem v obsluze:

$$E[W_S] = \frac{1}{\mu} \tag{90}$$

Střední doba strávená zákazníkem v systému:

$$E[W] = E[W_S] + E[W_Q] = \frac{1}{\mu} + \frac{E[N_Q]}{\lambda(1-P_Z)} \tag{91}$$

2.5 Systém M|D|1

V systému M|D|1 tvoří příchody zákazníků Poissonův proces s parametrem λ, ovšem doba obsluhy je konstantní, trvá $\frac{1}{\mu}$ pro každého zákazníka. Fronta není ohraničena. Nejedná se o Markovův proces a proto se pro řešení využívá metoda vnořených řetězců. [1, str. 42] Se způsobem řešení a odvozením vzorců se lze seznámit v [1, str. 43].

Předpokládá se zavedení pomocné proměnné ρ:

$$\rho = \frac{\lambda}{\mu} \tag{92}$$

Pravděpodobnost jednotlivých stavů systému: [1, str. 43]

$$p_n = \begin{cases} 1-\rho & \text{když } n = 0 \\ (1-\rho)(e^{\rho}-1) & \text{když } n = 1 \\ (1-\rho)\sum_{j=1}^{n-1}(-1)^{n-j}e^{j\rho}\left[\frac{(j\rho)^{n-j}}{(n-j)!}+\frac{(j\rho)^{n-j-1}}{(n-j-1)!}\right]+(1-\rho)e^{n\rho} & \text{když } n \geq 2 \end{cases} \tag{93}$$

Pravděpodobnost čekání zákazníka ve frontě:

$$P_Q = 1 - p_0 \tag{94}$$

Pravděpodobnost obsluhy zákazníka bez čekání:

$$P_S = p_0 \tag{95}$$

Střední počet zákazníků v systému: [1, str. 44]

$$E[N] = \frac{\rho(2-\rho)}{2(1-\rho)} \tag{96}$$

Střední počet zákazníků v obsluze:

$$E[N_S] = \frac{\lambda}{\mu} = \rho \tag{97}$$

Střední počet zákazníků ve frontě:

$$E[N_Q] = E[N] - E[N_S] = \frac{\rho^2}{2(1-\rho)} \tag{98}$$

Střední doba strávená zákazníkem ve frontě:

$$E[W_Q] = \frac{E[N_Q]}{\lambda} = \frac{\rho}{2\mu(1-\rho)} \tag{99}$$

Střední doba strávená zákazníkem v obsluze:

$$E[W_S] = \frac{1}{\mu} \tag{100}$$

Střední doba strávená zákazníkem v systému:

$$E[W] = \frac{E[N]}{\lambda} = \frac{2-\rho}{2\mu(1-\rho)} \tag{101}$$

2.6 Systém M|G|1

V modelu M|G|1 se předpokládá, že vstup zákazníků do systému tvoří Poissonův proces s parametrem λ. Systém obsahuje neohraničenou frontu a jednu linky obsluhy. Funkce hustoty pravděpodobnosti doby obsluhy na lince obsluhy není známa konkrétně, ale pouze je známa její střední hodnota $\frac{1}{\mu}$ a rozptyl σ^2. Náhodný proces není Markovův, pro řešení se používá metoda vnořených Markovových řetězců. [1, str. 44]. Odvození potřebných vzorců je možné najít např. v [5, str. 220].

Protože předpokládáme neznalost samotné funkce hustoty pravděpodobnosti doby obsluhy, nejsou pro tento model uváděny pravděpodobnostní charaktarestiky. V případě znalosti funkce hustoty pravděpodobnosti je ovšem získat lze a to na základě řešení soustavy rovnic. Podrobný postup je uveden např. v [1, str. 45].

Předpokládá se zavedení pomocné proměnné ρ:

$$\rho = \frac{\lambda}{\mu} \tag{102}$$

Střední počet zákazníků ve frontě: [5, str. 222]

$$E[N_Q] = \frac{\rho^2 + \lambda^2\sigma^2}{2(1-\rho)} \tag{103}$$

Střední počet zákazníků v obsluze:

$$E[N_S] = \frac{\lambda}{\mu} = \rho \tag{104}$$

Střední počet zákazníků v systému:

$$E[N] = E[N_Q] + E[N_S] = \frac{\rho^2 + \lambda^2\sigma^2}{2(1-\rho)} + \rho \tag{105}$$

Střední doba strávená zákazníkem ve frontě:

$$E[W_Q] = \frac{E[N_Q]}{\lambda} = \frac{\rho^2 + \lambda^2\sigma^2}{2\lambda(1-\rho)} \tag{106}$$

Střední doba strávená zákazníkem v obsluze:

$$E[W_S] = \frac{1}{\mu} \tag{107}$$

Střední doba strávená zákazníkem v systému:

$$E[W] = \frac{E[N]}{\lambda} = \frac{\rho^2 + \lambda^2\sigma^2}{2\lambda(1-\rho)} + \frac{1}{\mu} \tag{108}$$

Vzorce (103), (105), (106) a (108) se označují jako tzv. Chinčin-Pollaczekova formule. [5, str. 222]

3 SIMULAČNÍ ŘEŠENÍ

Simulační řešení přistupuje k řešení systémů hromadné obsluhy zcela jiným způsobem, než řešení analytické. Simulace spočívá v nahrazení skutečného systému jeho zjednodušeným modelem, se kterým se provádí experimenty. Experimenty jsou následně vyhodnocovány s vyvozováním určitých závěrů. Model systému může být v různých formách, např. fyzická zmenšenina (modelová železnice). Velmi často se ovšem využívá počítačového modelu, což je i případ této práce.

Počítačový model obsahuje struktury a algoritmy, které mohou být implementovány v libovolném obecném programovacím nebo speciálním simulačním jazyce. Pro tuto práci byl vybrán objektově orientovaný programovací jazyk Java.

Kapitola popisuje jednotlivé části procesu vývoje softwaru na jehož konci vznikla aplikace pro simulování některých typů systémů hromadné obsluhy.

3.1 Modelování a simulace

Pojmem modelování rozumíme cílevědomou činnost, která slouží k získávání informací o jednom systému prostřednictvím jiného systému — modelu. Mezi dvěma systémy může existovat jistá podobnost. Vzhledem k tomu, že model je také systémem, využívá se této podobnosti při modelování. Význam modelování spočívá v umožnění ekonomického studia systémů. Je výhodnější, rychlejší a často je jedinou možností získávat informace o systémech experimentováním s jejich modely, než originály. [11, str. 8]

Metody modelování systémů na počítačích jsou užitečné obzvláště v těchto případech: [11, str. 9]

- neexistuje úplná matematická formulace problému nebo nejsou známé analytické metody řešení,
- pokud jsou analytické metody dostupné pouze teoreticky a praktické řešení by bylo tak obtížné a dlouhé, že je metoda modelování problému na počítači jednodušší,
- modelování na počítači je jedinou možností získání výsledků v důsledku obtížnosti provádění experimentů ve skutečném prostředí,
- při pozorování systému je potřebné měnit časové měřítko — modelování na počítači umožňuje zrychlení nebo zpomalení příslušných dějů.

Pro účely modelování a simulací lze rozlišit tři základní časové pojmy: [11, str. 9]

1. *Reálný čas,* ve kterém probíhá skutečný děj v reálném systému.
2. *Modelový čas,* který tvoří časovou osu modelu — může být reálný, zrychlený nebo zpomalený.
3. *Strojový čas,* který představuje čas spotřebovaný na výpočet programu; jeho délka závisí na složitosti modelovaného systému a na vlastnostech vytvořeného programu.

Simulací označujeme etapu experimentování s reprezentací simulačního modelu. Jejím cílem je analýza chování systému v závislosti na vstupních veličinách a na hodnotách parametrů. [11, str. 11]

3.1.1 Generování náhodných veličin

Pro modelování náhodných veličin vyskytujících se v některých typech SHO se při simulaci využívá generátoru pseudonáhodných čísel. Stavebním blokem simulace je schopnost generovat hodnoty náhodné veličiny s rovnoměrným rozdělením v intervalu $\langle 0, 1)$. [12, str. 39]

Nejčastější metodou je lineární kongruentní generátor, kdy se posloupnost pseudonáhodných čísel získává na základě vztahu:

$$x_n = (ax_{n-1} + c) \mod m \tag{109}$$

kde a, c a m jsou vhodně zvolené konstanty. [12, str. 40]

Transformace rovnoměrného rozložení

Pro účely generování náhodných čísel s určitým rozdělením pravděpodobnosti je třeba náhodná čísla z intervalu $\langle 0, 1)$ transformovat na požadované rozdělení. Nejčastěji používanými metodami pro transformování náhodných čísel na zadané rozložení pravděpodobnosti jsou: [11, str. 97]

- metoda inverzní transformace,
- metoda vylučovací,
- metoda kompoziční.

Aplikace bude využívat exponenciálního rozdělení, pro jeho tvorbu je výhodná metoda inverzní transformace. Pro inverzní transformaci je třeba znát distribuční funkci rozdělení pravděpodobnosti. Platí, že náhodnou veličinu lze získat vztahem: [12, str. 70]

$$X = F^{-1}(U) \tag{110}$$

kde U je náhodná proměnná s rovnoměrným rozložením v intervalu $\langle 0, 1)$.

Ze vztahu (41) lze odvodit inverzní distribuční funkci pro určení náhodné proměnné s exponenciálním rozdělením pravděpodobnosti:

$$X = F^{-1}(U) = -\frac{1}{\lambda} \ln (1 - U) \tag{111}$$

3.1.2 Diskrétní simulace

Pro simulaci SHO lze s výhodou aplikovat diskrétní simulaci řízenou událostmi. Stav SHO se nemění souvisle, ale pouze v určitých časových okamžicích, proto není nutné kontinuálně sledovat stav systému, ale zaměřit se pouze na zmíněné události. [12, str. 111]

Implementace jádra diskrétní simulace využívá datovou strukturu „kalendář událostí“. Kalendář událostí podporuje dvě základní abstraktní operace: [11, str. 132]

1. Naplánování události na určitý čas.
2. Zjištění nejbližší události.

Algoritmus simulace poté vybírá z kalendáře vždy nejbližší událost, kterou provede. V rámci jejího provedení může nastat naplánování dalších událostí. Poté je zjištěna nejbližší událost v kalendáři a čas simulace je nastaven na její čas.

3.2 Analýza a specifikace požadavků

V prvotní fázi je třeba provést analýzu řešeného problému a specifikovat požadavky na vyvíjenou aplikaci. V případě této práce se bude jednat o desktopovou aplikaci, která bude umožňovat simulaci vybraných SHO.

Požadavky kladené na vyvíjenou aplikaci lze rozdělit na obecné požadavky, které jsou kladeny na aplikaci jako celek a poté na konkrétnější požadavky, kladené na její jednotlivé části. Použité rozdělení specifikace požadavků je následující:

- obecné požadavky na aplikaci,
- požadavky kladené na počítačový model SHO,
- požadavky kladené na GUI (grafické uživatelské rozhraní).

3.2.1 Obecné požadavky na aplikaci

1. Aplikace musí umožňovat vytvoření různých modelů SHO.
2. Aplikace musí umět provést simulaci vytvořeného modelu.

3. Aplikace zobrazuje výsledky simulace a vypočítává základní charakteristiky modelovaného SHO.
4. Aplikace je desktopového charakteru.
5. Aplikace obsahuje GUI.

3.2.2 Požadavky kladené na počítačový model SHO

1. Model může obsahovat různé komponenty.
2. Komponenty modelu jsou: zdroj, fronta, linka obsluhy, východ.
3. Komponenty modelu lze vzájemně propojovat.
4. Každá komponenta má unikátní název.
5. Každá komponenta má přidruženou svoji vlastní statistiku.
6. Zdroj podporuje deterministický příchod zákazníků.
7. Linka obsluhy podporuje deterministickou dobu obsluhy.
8. Zdroj podpoporuje náhodný příchod zákazníků s exponenciálním rozdělením pravděpodobnosti.
9. Linka obsluhy podporuje náhodnou dobu obsluhy s exponenciálním rozdělením pravděpodobnosti.
10. Fronta může být omezená s určenou kapacitou nebo neomezená.

3.2.3 Požadavky kladené na GUI

1. GUI umožňuje tvorbu modelu přidáváním různých modelových komponent.
2. GUI umožňuje vybrat komponentu a změnit její nastavení.
3. GUI umožňuje tvorbu a úpravu vazeb mezi komponentami.
4. GUI umožňuje mazat komponenty v modelu.
5. GUI vizualizuje graficky komponenty a graficky znázorňuje jejich vzájemné propojení v modelu.
6. GUI umožňuje přemisťovat grafické komponenty na plátně.
7. GUI umožňuje spustit simulaci.
8. GUI umožňuje zastavit probíhající simulaci.
9. GUI umožňuje zobrazit statistiku jednotlivých komponent.
10. GUI zobrazuje statistiku v reálném čase, tak jak je počítána během probíhající simulace.

3.3 Návrh aplikace

Aplikace bude vytvořena v programovacím jazyce Java. Jazyk Java byl vybrán z několika důvodů, mezi které patří:

- objektově orientovaný přístup,
- podpora tvorby GUI,
- multiplatformost (včetně multiplatformního GUI),
- univerzálnost jazyka (jedná se o obecný programovací jazyk),
- otevřenost technologie (OpenJDK),
- automatická správa paměti (vestavěný Garbage Collector).

Výše uvedené vlastnosti splňují pravděpodobně i mnohé jiné programovací jazyky.

Aplikace je založena na modulárním návrhu, přičemž ji lze rozdělit do třech základních modulů:

- modul simulačního modelu,
- modul statistiky,
- modul GUI.

Pro moduly je charakteristické, že modul simulačního modelu a statistiky může fungovat zcela odděleně a nezávisle na modulu GUI, což umožnuje využít aplikační programové rozhraní (API) pro využití v jiných programech, či umožňuje tvorbu jednoduchých jednoúčelových programů pro příkazovou řádku (CLI).

3.4 Modul simulačního modelu

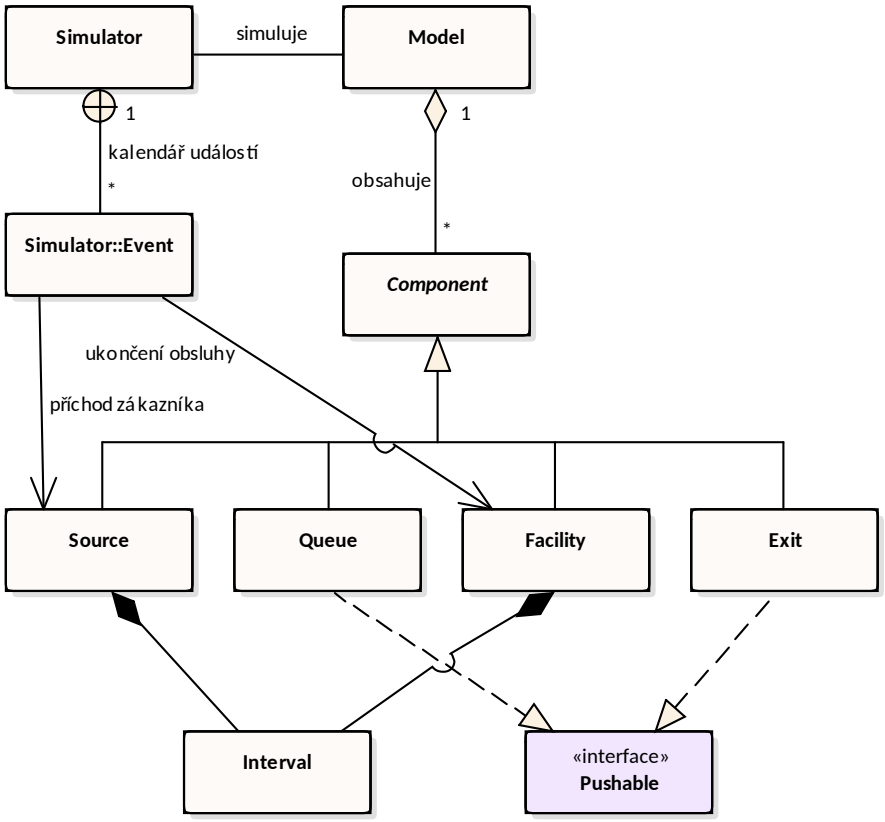
Modul simulačního modelu tvoří samotné jádro celé aplikace. Obsahuje třídy, které reprezentují různé komponenty SHO. Pomocí těchto komponent lze sestavovat různé modely SHO. Kromě základních komponent modelu modul obsahuje samotný simulátor, který provádí simulaci chování daných komponent v čase.

3.4.1 Návrh tříd a implementace

Modul je implementován v balíčku `qs.model`. Z různých tříd a rozhraní, které jsou v balíčku implementovány, je nezbytné zmínit alespoň tyto následující:

- třída `Component`,

- třída `Source`,
- třída `Queue`,
- třída `Facility`,
- třída `Exit`,
- třída `Model`,
- třída `Simulator`,
- rozhraní `Pushable`.



Obrázek 14: Diagram vybraných tříd z balíčku qs.model. Zpracování autora.

Obrázek 14 znázorňuje důležité vazby mezi vybranými třídami v balíčku `qs.model`. Třída `Model` je představitelem samotného modelu systému hromadné obsluhy. Obsahuje libovolné množství různých komponent, které jsou do modelu uživatelem přidány. Komponenta je vyjádřena abstraktní třídou `Component`, přičemž existují 4 potomci této třídy: `Source` (zdroj), `Queue` (fronta), `Facility` (linka obsluhy) a `Exit` (východ).

Samotný model i komponenty v balíčku `qs.model` obsahují pouze logické vlastnosti modelu, neobsahují žádné informace o chování grafických komponent, kterými jsou repre-

zentovány v grafickém uživatelském rozhraní aplikace. Grafická nadstavba těchto komponent je obsažena v balíčku `qs.gui`.

Komponenty (přesněji řečeno potomci třídy `Component`) mají pouze aktuální stav v určitém časovém okamžiku simulace a neznají žádné informace z minulosti nebo budoucnosti. Pro lepší představu uvažujme např. třídu `Facility` — tato třída v rámci své zodpovědnosti zná pouze to, zdali je v její obsluze nějaký zákazník (popř. který konkrétní zákazník), či zdali je linka obsluhy prázdná. Samotná třída ovšem nijak neeviduje, kolik zákazníku již bylo linkou obsluhy obslouženo (minulost), ani nezná okamžik, kdy bude obsluha zákazníka dokončena (budoucnost). Podobná situace je i u ostatních komponent. Evidenci statistických údajů má na starosti samotný modul statistiky. O plánování okamžiků ukončení obsluhy a vzniku nových zákazníků zase rozhoduje simulační jádro, reprezentované třídou `Simulator`.

Třídy `Queue` a `Exit` narozdíl od ostatních komponent implementují rozhraní `Pushable`. Zmíněné rozhraní zaručuje, že komponenta je schopna vždy na vstupu přijmout zákazníka, nehledě na vnitřní stav komponenty. Tuto vlastnost zaručuje východ a fronta. Oproti tomu zdroj zákazníků žádný vstup zákazníků neakceptuje. Linka obsluhy sice vstup zákazníků umožňuje, ale jen za podmínky, že již právě neobsluhuje jiného zákazníka.

Uzamykatelnost modelu

Model i samotné komponenty obsahují mechanismus pro uzamknutí jejich stavu. Jakmile simulátor zahájí simulaci, pomocí metody `lock()` uzamkne model, což se kaskádovitě projeví i na uzamknutí obsažených komponent. Pokud je model v uzamknutém stavu, nelze přidávat ani odebírat žádné komponenty. U komponent nelze měnit názvy, parametry a ani vazby mezi vstupy a výstupy jednotlivých komponent.

Na konci simulace simulátor voláním metody `unlock()` odemkne model a jeho komponenty pro další úpravy. Mechanismus uzamknutí zaručuje, že po celou dobu provádění simulace nelze měnit parametry a složení komponent v modelu (např. paralelně z jiných vláken) a zaručuje, že po celou dobu simulace bude model v konzistentním a validním stavu. Oboje zmíněné metody mají modifikátor přístupu `protected` a proto stav uzamčení modelu a komponent nemůže být měněn uživatelem, či jinými moduly aplikace.

Události v modelu

Komponenty ani model neuchovávají historii svého stavu, ale poskytují flexibilní mechanismus, který umožňuje sledování stavu jiným třídám. Princip je inspirován návrhovým vzorem „Observer“. Na modelu i komponentách lze pomocí metody `addListener()` registrovat objekt, který bude notifikován v případě výskytu události v modelu, či na komponentě. V průběhu simulace, jsou registrovanému objektu předávány informace o událostech, které se v modelu či na komponentách vyskytly. Součástí informace o události je:

- aktuální čas simulace, ve kterém událost nastala,
- zákazník, jehož se událost týká,
- konkrétní komponenta, na které se událost vyskytla,
- typ události.

Přičemž je rozlišováno 7 následujících typů událostí:

- zdroj — příchod nového zákazníka do systému,
- fronta — přijmutí zákazníka do fronty,
- fronta — odmítnutí zákazníka frontou,
- fronta — odchod zákazníka z fronty,
- linka obsluhy — začátek obsluhy zákazníka,
- linka obsluhy — konec obsluhy zákazníka,
- východ — odchod zákazníka ze systému.

Protože je celý mechanismus událostí v modelu koncipován takto obecně, může mít různá využití — např. může posloužit k tvorbě protokolu o vzniklých událostech při simulaci modelu. V aplikaci je ovšem tohoto principu využito modulem statistiky, kdy jsou na jednotlivé komponenty modelu navěšeny statistické třídy, které sledují a zaznamenávají charakteristické veličiny komponent.

Třída Model

Třída `Model` sdružuje všechny komponenty do jednoho celku, na kterém je poté prováděna simulace. Poskytuje tedy především metody pro přidání nebo odebrání jednotlivých komponent. Poskytuje kaskádový mechanismus uzamykání, kdy při uzamknutí modelu uzamkne všechny obsažené komponenty, stejně tak i při odemknutí.

Obsahuje metodu `addListener()`, která registruje objekt, který je notifikován o jakékoliv události, která se vyskytne na jakékoliv komponentě v modelu. Tato funkcionalita

je totožná tomu, pokud bychom stejný objekt registrovali na všech komponentách obsažených v modelu.

Třída Component

Třída `Component` představuje abstraktní třídu, která je předchůdcem všech základních komponent modelu (zdroj, fronta, linka obsluhy a východ). Třída implementuje rysy a chování, které je společné pro všechny prvky modelu. Kompletní přehled všech atributů a metod zobrazuje obrázek 15.

Implementuje společné rysy mechanismu pro uzamykání komponent. Definuje metody `isLocked()`, `unlock()` a předepisuje abstraktní metodu (tzn. každý potomek ji musí přepsat vlastní implementací) `lock()`. Metoda `lock()` není implementována z toho důvodu, protože při uzamčení komponenty probíhá i její logická validace (např. zdroj musí mít nastaven výstup, apod.), což je pro každý druh komponenty její specifický proces.

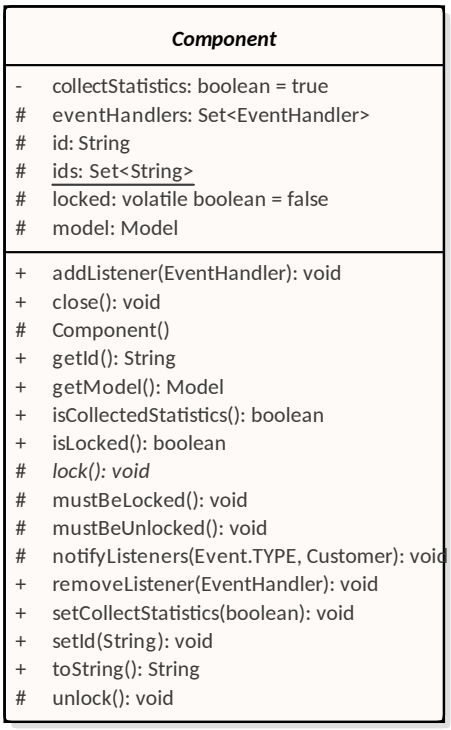
Mezi další společné rysy patří popisovaný mechanismus vzniku událostí na jednotlivých komponentách. Ačkoliv vznik událostí je specifický pro každý druh komponenty, třída `Component` implementuje metody pro registraci a odregistraci objektů — tj. metody `addListener()` a `removeListener()`. Svým potomkům poskytuje interní metodu `notifyListeners()`, díky které jednoduše předávají informaci o vzniku události všem objektům, které mají zájem být o události informováni.

U každé komponenty je možné nastavit příznak, zdali má být pro komponentu vytvářena statistika. Za tímto účelem slouží metoda `setCollectStatistics()`. Standardně je pro každou komponentu ve výchozím nastavení modulem statististiky sbírána a vytvářena její statistika, což ovšem nemusí být vždy vyžadováno. Vypnutí statistiky komponenty má pozitivní vliv na rychlost a paměťové nároky provedení simulace.

Poslední (avšak neméně důležitou) zodpovědností třídy `Component` je poskytování metod `getId()` a `setId()` pro jednotlivé komponenty, přičemž je kontrolována a zaručována unikátnost jmen, tak aby jméno všech existujících komponent bylo unikátní a např. potomek `Facility` (linka obsluhy) nemohl užívat stejný identifikátor jako jiný druh komponenty — např. `Queue` (fronta).

Třída Source

Třída `Source` představuje komponentu pro zdroj nových zákazníků. Komponentě lze nastavit interval, v jakém zákazníci mají přicházet do systému. Specifikace intervalu je v režii



Obrázek 15: Vizualizace třídy Component. Zpracování autora.

třídy `Interval` a může být deterministický nebo náhodný s exponenciálním rozdělením pravděpodobnosti. Nastavení se provádí buď definováním střední hodnoty nebo intenzity.

Komponenta nemá žádný vstup z ostatních komponent, ale má výstup, který musí být napojen na komponentu implementující rozhraní `Pushable` (tj. fronta nebo východ). Nastavení výstupu se provádí metodou `setOutput()`. Při uzamknutí komponenty je kontrolováno, zdali je výstup opravdu nastaven.

Pro simulátor poskytuje třída interní metodu `generate()`. Metoda vytváří instanci nového zákazníka, kterého bezprostředně po vyvolání události vzniku nového zákaznika předá komponentě napojené na výstup, zavoláním její metody `arrival()`.

Třída Queue

Třída `Queue` (fronta) je komponentou umožňující shlukování a čekání zákazníků v modelu na jejich obsluhu. Fronta pracuje v režimu FIFO. Umožňuje nastavit, zdali bude mít omezenou či neomezenou[4] kapacitu. Lze též nastavit výchozí počet zákazníků, kteří jsou ve frontě ihned po zahájení simulace.

Pro propojení s ostatními komponentami má fronta jeden vstup a dva výstupy. Vstup komponenty může být napojen na výstup jakékoliv jiné komponenty. Oproti tomu běžný

[4]I v případě neomezené kapacity je její velikost omezena implementačně datovým typem integer.

výstup fronty může být spojen pouze s linkou obsluhy, která si z fronty „vytahuje“ zákazníky voláním interní metody `pickup()`. Druhý výstup se používá pouze v případě fronty s omezenou kapacitou a jedná se o výstup pro případ, kdy je zákazník frontou odmítnut. Tento výstup musí směřovat na komponentu implementující rozhraní `Pushable`.

Příchod zákazníka do fronty je signalizován komponentou, kterou zákazník opouští pomocí volání interní metody `arrival()`. Fronta na základě své maximální kapacity a aktuální obsazenosti rozhodne, zdali zákazníka přijme, či předá výstupnímu prvku pro odmítnuté zákazníky. Pokud je tímto prvkem opět fronta (ať už ta stejná, či jiná), mohlo by dojít k zacyklení zákazníka mezi frontami. Za tímto účelem je stanoven maximální limit (100), udávající maximalní počet front, kterými může zákazník bezprostředně za sebou projít, aby se zabránilo potenciálnímu zacyklení simulace.

Při uzamknutí komponenty dochází ke kontrole nastavení výstupního prvku pro odmítnuté zákazníky, který musí být nastaven vždy, pokud se jedná o frontu s omezenou kapacitou. Dále dochází k vytvoření instancí jednotlivých zákazníku, pokud má fronta nastaven nenulový počet výchozích zákazníků.

Třída Facility

Komponenta linky obsluhy, která je představována třídou `Facility` má za úkol simulovat obsluhu jednoho zákazníka po určitý časový interval. Doba obsluhy je vyjádřena třídou `Interval`, jsou tedy podporovány stejné funcionality jako v případě intervalu příchodu zákazníků u komponenty zdroj.

Linka obsluhy má jeden vstup, kterým musí být fronta a jeden výstup, kterým je komponenta implementující rozhraní `Pushable`. Při uzamknutí komponenty je kontrolováno, zdali existuje správný vstupní i výstupní prvek. Simulátoru jsou k dispozici 2 interní metody, a to `beginService()` a `endService()`, které zahajují a ukončují obsluhu zákazníka na lince obsluhy. Metoda `beginService()` vrací návratovou hodnotu typu `boolean`, která značí, zda skutečně začla obsluha zákazníka (může se stát, že ve frontě nikdo nečeká a není tedy koho obsluhovat).

Třída Exit

Třída `Exit` představuje komponentu východu, přičemž se jedná o funkčně nejjednodušší komponentu. Obsahuje pouze jediný vstup a interní metodu `arrival()`, která přijme

zákazníka (instanci třídy `Customer`). O této skutečnosti je vygenerovaná událost a pokud nejsou udržovány jiné reference na zákazníka, tak objekt zákazníka zaniká.

Třída Simulator

Samotné jádro simulátoru je tvořeno třídou `Simulator`. Třída má k dispozici instanci modelu, na kterém se simulace provádí. Protože je simulace založena na přístupu diskrétní simulace řízené událostí, uchovává simulátor instanci kalendáře událostí, aktuální modelový čas, který řídí a koncový čas, v němž má být simulace ukončena.

Jakmile je pomocí metod `setModel()` a `setEndTime()` přiřazen model SHO, kterého se simulace týká a konečný čas simulace, je zapotřebí pomocí metody `init()` provést inicializaci běhu simulace. V tomto inicializačním kroku se do kalendáře událostí naplánují časy vygenerování prvních zákazníků ve zdrojích a provede se případné zahájení činnosti linek obsluhy, pokud fronty nejsou v nulovém čase simulace prázdné.

Po prvotní inicializaci již následuje samotný běh simulace, který je uskutečňován voláním metody `step()` tak dlouho, dokud je její návratová hodnota logické hodnoty `true`. S každým zavoláním se provede jeden krok simulace, což odpovídá provedení první nejbližší události z kalendáře událostí.

Vybere se nejbližší událost, pokud je kalendář událostí prázdný simulace končí. Simulátor nastaví modelový čas simulace na naplánovaný čas události a podle druhu události volá patřičné metody na komponentách v modelu. Existují dva druhy naplánovaných událostí:

- příchod nového zákazníka,
- ukončení obsluhy zákazníka na lince obsluhy.

Při příchodu nového zákazníka je požádán příslušný zdroj o vygenerování nového zákazníka. Ten je poslán na výstupní prvek zdroje, kde se zákazníka ujme dotyčná výstupní komponenta. Do kalendáře událostí je naplánována příští událost příchodu nového zákazníka na daném zdroji, přičemž se požádá třída `Interval` o vygenerování nové (ať už deterministické nebo stochastické) hodnoty, která je přičtena k aktuálnímu modelovému času.

Pokud se jedná o událost ukončení obsluhy zákazníka na lince obsluhy, příslušná linka obsluhy je o této skutečnosti informována zavoláním její metody `endService()`. Zákazník je poté předán výstupní komponentně linky obsluhy.

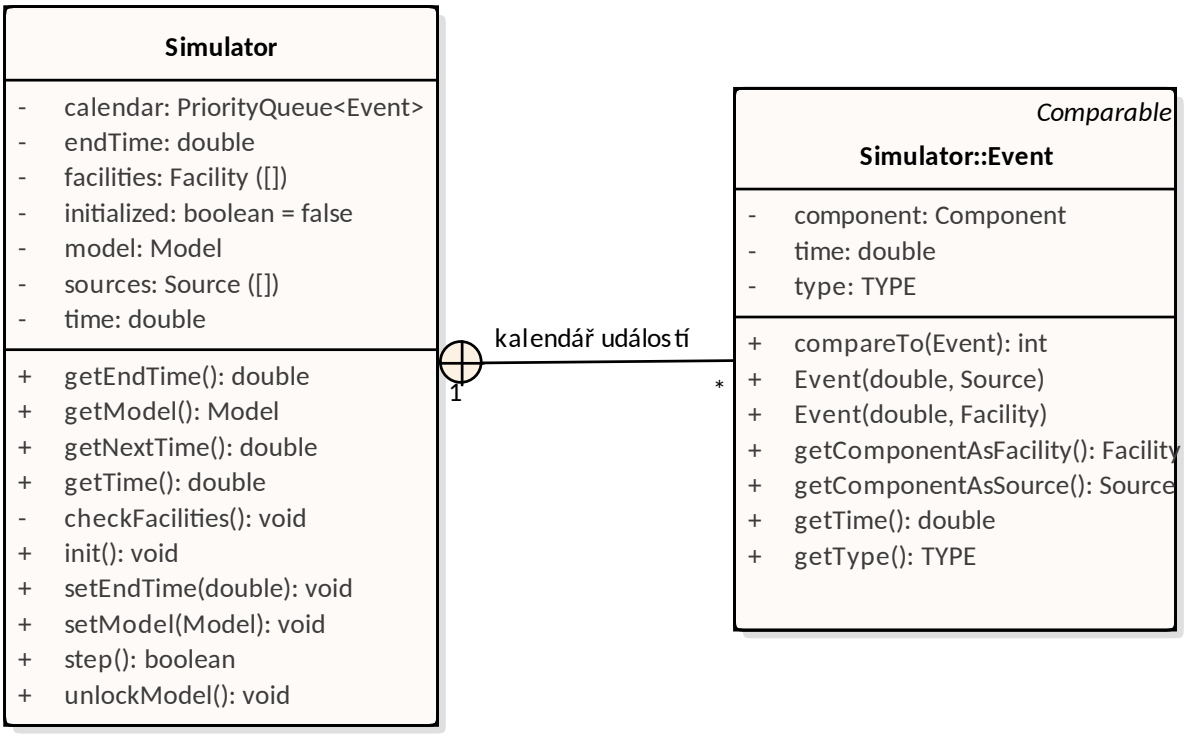
Při obou dvou zmíněných událostech jsou zkontrolovány linky obsluhy. Pokud je některá z linek obsluhy prázdná a existuje čekající zákazník ve frontě, ihned začne jeho

obsluha. Při této události se naplánuje do kalendáře událostí čas ukončení obsluhy na příslušné lince.

Vnořená třída `Event` vyjadřuje položku kalendáře událostí. Tato položka obsahuje:

- typ události,
- čas plánovaného spuštění události,
- komponentu, které se událost týká (zdroj nebo linka obsluhy).

Kompletní přehled všech atributů a metod třídy `Simulator` a vnořené třídy `Event` zobrazuje obrázek 16.



Obrázek 16: Vizualizace třídy Simulator. Zpracování autora.

3.4.2 Ukázka použití modulu simulačního modelu

Následující příklad demonstruje využití modulu simulačního modelu napřímo v Java programu. Předpokladem je M|D|2 systém, kdy intenzita příchodů zákazníků do systému tvoří Poissonův proces s intenzitou 6 zákazníků/hodinu. Systém obsahuje neomezenou frontu a dvě linky obsluhy, přičemž doba obsluhy trvá 0,25 hodiny (15 minut). Celkový modelový čas simulace je 48 hodin.

Vytvoříme instance jednotlivých komponent:

```
Source s1 = new Source();
Queue q1 = new Queue();
Facility f1 = new Facility();
Facility f2 = new Facility();
Exit e1 = new Exit();
```

Specifikujeme interval příchodů a dobu obsluhy:

```
Interval prichody = new Interval();
prichody.setType(TYPE.EXPONENTIAL);
prichody.setRate(5.0);
s1.setInterval(prichody);
Interval dobaObsluhy = new Interval();
dobaObsluhy.setType(TYPE.DETERMINISTIC);
dobaObsluhy.setExpectedValue(0.25);
f1.setInterval(dobaObsluhy);
f2.setInterval(dobaObsluhy);
```

Propojíme komponenty:

```
s1.setOutput(q1);
f1.setInput(q1);
f1.setOutput(e1);
f2.setInput(q1);
f2.setOutput(e1);
```

Následně přidáme komponenty do modelu, nastavíme čas simulace a spustíme:

```
Model m = new Model();
m.add(s1);
m.add(q1);
m.add(f1);
m.add(f2);
m.add(e1);
Simulator sim = new Simulator();
sim.setEndTime(48.0);
```

```
sim.setModel(m);
sim.init();
while(sim.step());
```

3.5 Modul statistiky

Ačkoliv je modul simulačního modelu jádrem celého programu, jeho samostatné využití nemá moc význam. Simulace sice proběhne přesně podle specifikace, ale neexistuje žádný smysluplný výstup programu. Za tímto účelem slouží model statistiky (balíček `qs.statistics`), který umožňuje zaznamenat důležité události v průběhu simulace a vypočítat statistické charakteristiky systému.

Standardně obsahuje každá komponenta svoji statistiku, která bude dále stručně popsána. Uživatel není odkázán na statistické třídy v tomto modulu, ale díky mechanismu informování o událostech v modelu si může vytvořit implementaci vlastní statistiky.

Statistika modelu

Statistika modelu je implementována třídou `ModelStatistics` a vztahuje se k celému modelu. V rámci statistiky jsou implementovány jednoduché čítače, které evidují počet příchozích a odchozích zákazníků v modelu. Jedná se o souhrnná data ze všech komponent typu zdroj nebo východ. Při sledování počtu zákazníků v systému se rovněž zaznamenává minimum a maximum.

Metoda `getProbabilityState(int n)` vrací pravděpodobnost, že v systému je právě n zákazníků. Tato pravděpodobnost je statistického charakteru a je spočtena jako:

$$p_n = \frac{t_n}{T} \tag{112}$$

kde t_n je čas strávený ve stavu n (tj. celkový čas, který bylo v systému právě n zákazníků) a T je aktuální modelový čas simulace. Za tímto účelem je interně udržována tabulka strávených časů, kde jsou uchovávány časy t_n pro jednotlivé stavy n.

Střední hodnota času stráveného zákazníkem v systému je spočtena podílem:

$$E[W] = \frac{\text{celkový čas všech zákazníků v systému}}{\text{celkový počet všech zákazníků v systému}} \tag{113}$$

Střední počet zákazníků v systému je spočten:

$$E[N] = \sum_{n=a}^{b} p_n n \tag{114}$$

kde a je minimální a b maximální počet zákazníků v systému.

Statistika zdroje a východu

Jedná se o jednodušší statistiky, které pouze obsahují čítač skutečného počtu příchozích/odchozích zákazníků a jsou schopny určit společně s použitím aktuálního modelového času simulace skutečnou střední hodnotu intenzity příchodů/odchodů, popř. intervalu mezi příchody/odchody.

Statistika fronty

Statistika fronty je podobná statistice modelu. Jednotlivé čítače evidují počty čekajících, průchozích[5] a odmítnutých zákazníků. Na základě jednotlivých četností jsou stanoveny pravděpodobnosti čekání, průchodu a odmítnutí.

Při využití stejných vztahů a principů jako v případě statistiky modelu je evidován minimální a maximální počet zákazníků ve frontě, vypočítávána střední délka fronty a střední doba čekání zákazníků ve frontě. Obdobně jako jsou vypočítávány pravděpodobnosti jednotlivých stavů systému jsou i vypočítávány pravděpodobnosti jednotlivých stavů fronty.

Statistika linky obsluhy

Statistika linky obsluhy eviduje počet obsloužených zákazníků a celkový pracovní čas. Na základě těchto údajů je spočteno vytížení linky obsluhy a jsou získány střední hodnoty skutečného času stráveného zákazníky v obsluze a skutečná střední hodnota intenzity obsloužených zákazníků.

3.6 Modul GUI

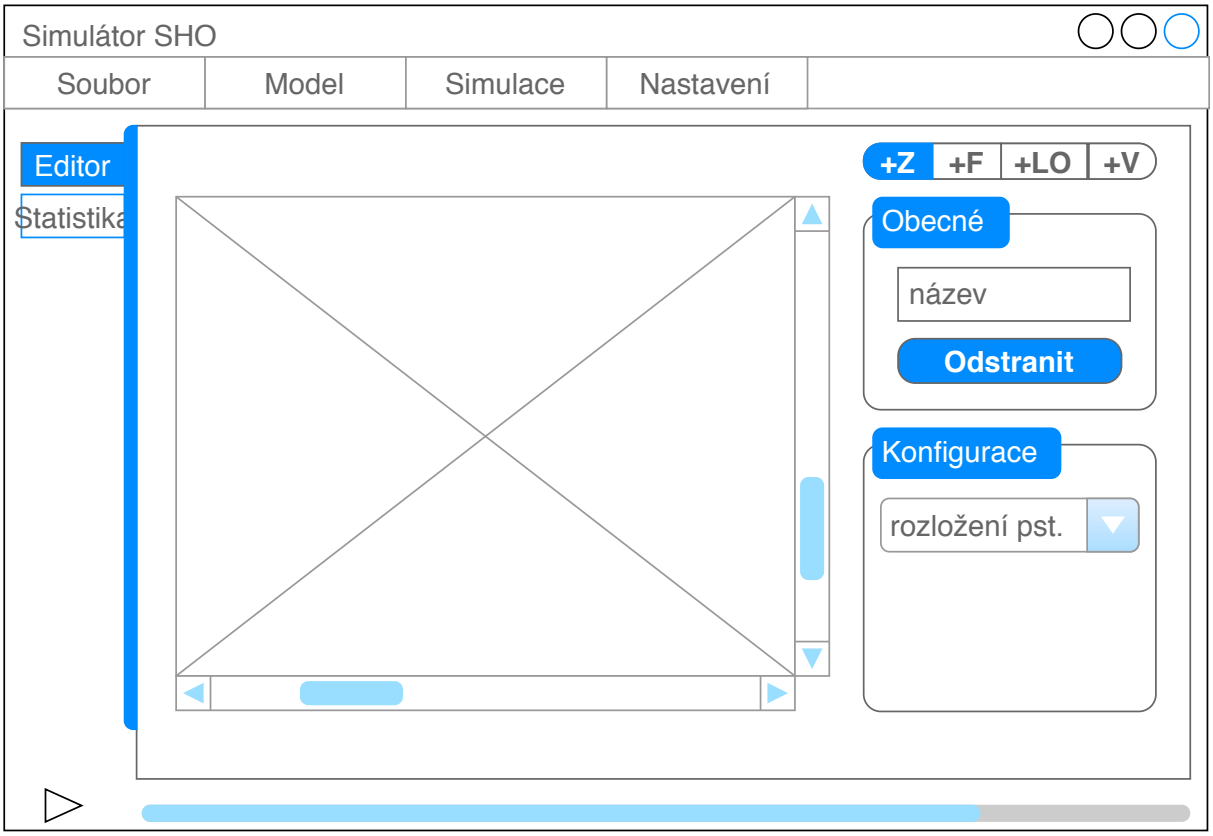
Pro podporu uživatelské přívětivosti a zefektivnění tvorby simulačních modelů je součástí aplikace modul s grafickým uživatelským rozhraním. Tento modul využívá multiplatformní grafickou knihovnu JavaFX. Modul GUI je implementován v balíčku `qs.gui`.

3.6.1 Návrh GUI

Na obr. 17 je znázorněn jednoduchý náčrt návrhu GUI. V návrhu je obsaženo rozvržení základních grafických komponent. Aplikace by měla mít 2 základní funkční módy —

[5] Výrazem „průchozí“ jsou označeni zákazníci, kteří jsou frontou přijati, ale stráví v ní nulový čas, protože ihned pokračují na linku obsluhy.

první umožňuje vytváření a editaci modelu, druhý umožňuje sledovat statistické veličiny jednotlivých komponent v průběhu simulace.



Obrázek 17: Návrh grafického uživatelského rozhraní. Zpracování autora.

Editační mód by měl umožňovat přidávat do modelu nové komponenty. Tyto komponenty mohou být pomocí myši různě přemisťovány v rámci pracovní plochy. Po výběru konkrétní komponenty by měl jít nastavit její název a další charakteristické hodnoty. Na pracovní ploše by se měla zobrazovat propojení vstupů a výstupů jednotlivých komponent. Propojení mohou být uživatelem konfigurována. Pracovní plocha by měla být posuvná a neměla by uživatele omezovat svojí velikostí.

Uživatel má k dispozici hlavní menu, které by mělo sloužit k ostatním akcím (např. nastavení parametrů simulace, uložení modelu apod.). Ve spodní části okna je umístěno tlačítko pro spuštění simulace a grafické znázornění průběhu simulace.

Mód statistika slouží pro procházení a zobrazování statistik jednotlivých komponent. Uživatel má k dispozici pracovní plochu podobnou jako v editačním módu. Z této pracovní plochy vybírá komponentu, jejíž podrobnou statistiku si chce zobrazit. Statistika by měla být vypočítávána interaktivně i v průběhu běhu simulace.

3.6.2 Implementace GUI

Základní grafické rozvržení aplikace a rozmístění konkrétních ovládacích prvků je provedeno pomocí aplikace SceneBuilder, která slouží ke grafickému návrhu aplikací využívajících JavaFX. Výstupem aplikace je XML soubor, který představuje stromovitou strukturu jednotlivých grafických komponent, popisuje způsob jejich rozvržení a konfiguruje atributy konkrétních komponent. K této stromové struktuře je připojen tzv. controller, což je třída v jazyce Java, starající se o funkčnost a interakci jednotlivých prvků. V případě aplikace „Simulátor SHO“ je hlavním controllerem GUI třída `MainWindowController`.

Třída MainWindowController

Třída `MainWindowController` představuje jednu z nejrozsáhlejších tříd v modulu GUI. Jejím úkolem je správně propojit všechny grafické prvky v aplikaci, reagovat na události uživatele (kliknutí na tlačítko, táhnutí myší, stisk klávesy apod.), reagovat na události některých grafických prvků, udržovat v paměti grafickou reprezentaci simulačního modelu i s případně nastavenými atributy a mnohé další.

Třídy GraphicsModel, GraphicsComponent, GraphicsSource, ...

Třídy začínající svůj název řetězcem `Graphics` obvykle reprezentují grafickou formu komponent simulačního modelu. Každá z tříd vždy obaluje příslušnou třídu z balíčku `qs.model` a přidává prvky nutné pro grafické vyobrazení.

Jednotlivé grafické komponenty dědí z abstraktní třídy `GraphicsComponent`, která obsahuje několik různých metod, které musí potomci implementovat. Mezi předepsané metody patří:

1. Metody pro podporu získání a nastavení souřadnic, na kterých je komponenta vykreslována.
2. Metody pro zjištění výšky a šířky grafiky dané komponenty.
3. Metody pro získání absolutních souřadnic vstupních a výstupních bodů komponent.
4. Metoda `render()` pro grafické vyobrazení komponenty na nastavených souřadnicích.

Třída CanvasController

Třída má na starosti propojení simulačního modelu (který je v reprezentován třídou `GraphicsModel`) a samotného plátna pro vykreslování grafiky, které je součástí knihovny JavaFX a je reprezentováno třídou `Canvas`.

`CanvasController` udržuje svůj interní stav, který může nabývat tří hodnot: `BASE`, `SELECTED` a `MOVING`. Stav `BASE` je stavem základním a v tomto stavu se objekt nachází po spuštění aplikace. Ve stavu `BASE` proběhne vykreslení grafického modelu a poté jsou sledovány události myši. Při pohybu myši je zjišťováno, zdali se pod kurzorem nenachází nějaká grafická komponenta. Pokud nachází, tato komponenta je vykreslena zvýrazněnou barvou. Pokud dojde ke stisknutí tlačítka myši nad komponentou grafického modelu, tato komponenta je označena jako vybraná a `CanvasController` přechází do stavu `SELECTED`. Při přechodu do tohoto stavu je vyvolána událost informující o změně vybrané komponenty. Na událost reaguje třída `MainWindowController` aktualizací formulářových prvků pro konfiguraci komponenty. Posledním stavem je `MOVING`, který nastává při detekovaném táhnutí komponenty myší. V tomto případě je komponentě periodicky měněna její pozice dle souřadnic kurzoru myši. Výsledkem je přemísťování komponenty.

Při pohybu komponent po plátně jsou automaticky zjišťovány hraniční komponenty (tj. komponenta umístěná nejvíce vpravo a nejvíce dole), dle kterých jsou dynamicky upravovány rozměry plátna. Pokud plátno přesáhne oblast vyhrazenou pro zobrazení jsou zobrazeny posuvníky.

Komponenty jsou na plátně vykreslovány v pravidelném intervalu 40 ms, což odpovídá zobrazení 25 snímků/s. Aby nedocházelo k neustálému překreslování (a zátěži CPU), poskytuje třída `CanvasController` mechanismum, který překresluje plátno pouze po detekované změně v grafickém modelu nebo aktivním pohybu myší nad plátnem.

Třída `CanvasController` sleduje vzájemné propojení vstupů a výstupů jednotlivých komponent v modelu. Na základě zjištěných propojení jsou vstupní a výstupní body příslušných komponent na plátně graficky spojeny přímou čárou.

Třída SimulationTask

Reprezentuje úlohu provádění samotné simulace modelu. S třídou `SimulationTask` komunikuje `MainWindowController`, který po stisku patřičného tlačítka uživatelem, spouští nebo zastavuje simulaci. Simulace je spuštěna v novém vlákně, aby svým náročným výpočtem a možným dlouhým trváním nenarušovala vykreslování a odezvu GUI. Prostřed-

nictvím třídy `SimulationTask` lze zjišťovat stav průběhu simulace, který je uživateli interaktivně vizualizován. V průběhu simulace není možné upravovat simulační model, aby nemohlo dojít k neočekávaným, či zkresleným výsledkům.

Třída StatisticsThread

Reprezentuje vlákno, které má na starosti výpočet statistických charakteristik modelu a to paralelně s běžící simulací. Jakmile je uživatelem spuštěna simulace ve statistickém vlákně, začne v periodických intervalech probíhat výpočet statistických charakteristik vybrané komponenty. Třídou `MainWindowController` je dotazováno statistické vlákno každých 100 ms. Chování se liší v závislosti na interním stavu vlákna, které může být:

- ve stavu připraveno,
- ve stavu probíhající,
- ve stavu dokončeno.

Pokud je vlákno ve stavu připraveno, tak je požádáno o výpočet statistiky konkrétní komponenty (dle toho, kterou má uživatel vybranou). Po uplynutí časového intervalu je statistické vlákno opět dotázáno, přičemž jeho stav může být dokončeno nebo probíhající.

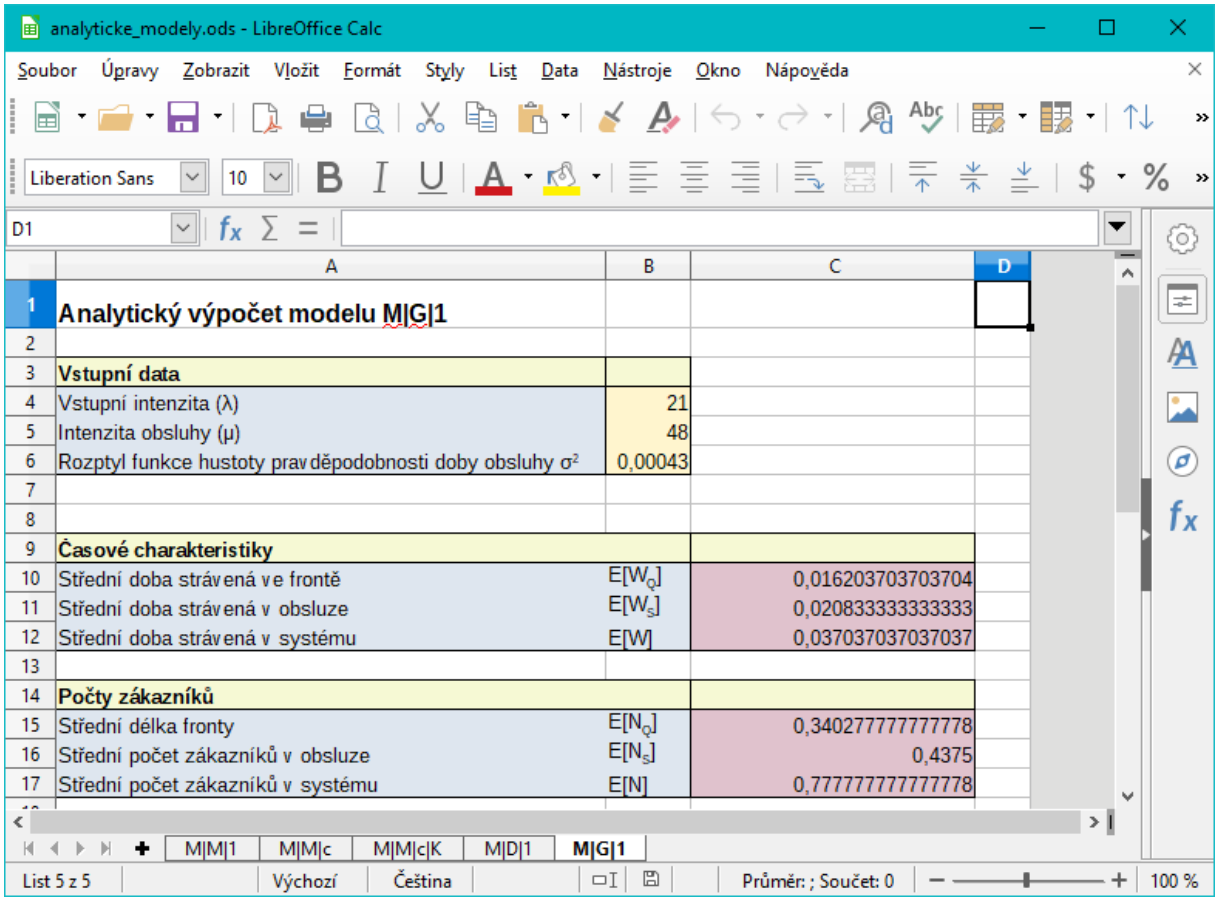
1. Pokud je vlákno ve stavu dokončeno, převezmou se z vlákna vypočítané charakteristiky, které se zobrazí uživateli a vláknu je zadán nový požadavek.
2. Jestliže vlákno nestihlo dokončit svůj výpočet a je stále ve stavu probíhající, záleží zdali uživatel nezměnil svůj výběr komponenty. Pokud nezměnil, vlákno pokračuje ve výpočtu statistiky. V opačném případě je přerušeno a je mu zadán nový požadavek.

Interaktivní výpočet statistických charakteristik umožňuje uživateli „živě“ sledovat, jak se jednotlivé veličiny v průběhu simulačního času vyvíjí.

4 SROVNÁNÍ ANALYTICKÉHO A SIMULAČNÍHO ŘEŠENÍ

V následující části práce bude provedeno srovnání analytického a simulačního přístupu při řešení běžných modelů THO. Pro srovnání je využito modelů, jejichž analytické řešení bylo představeno v kapitole 2.

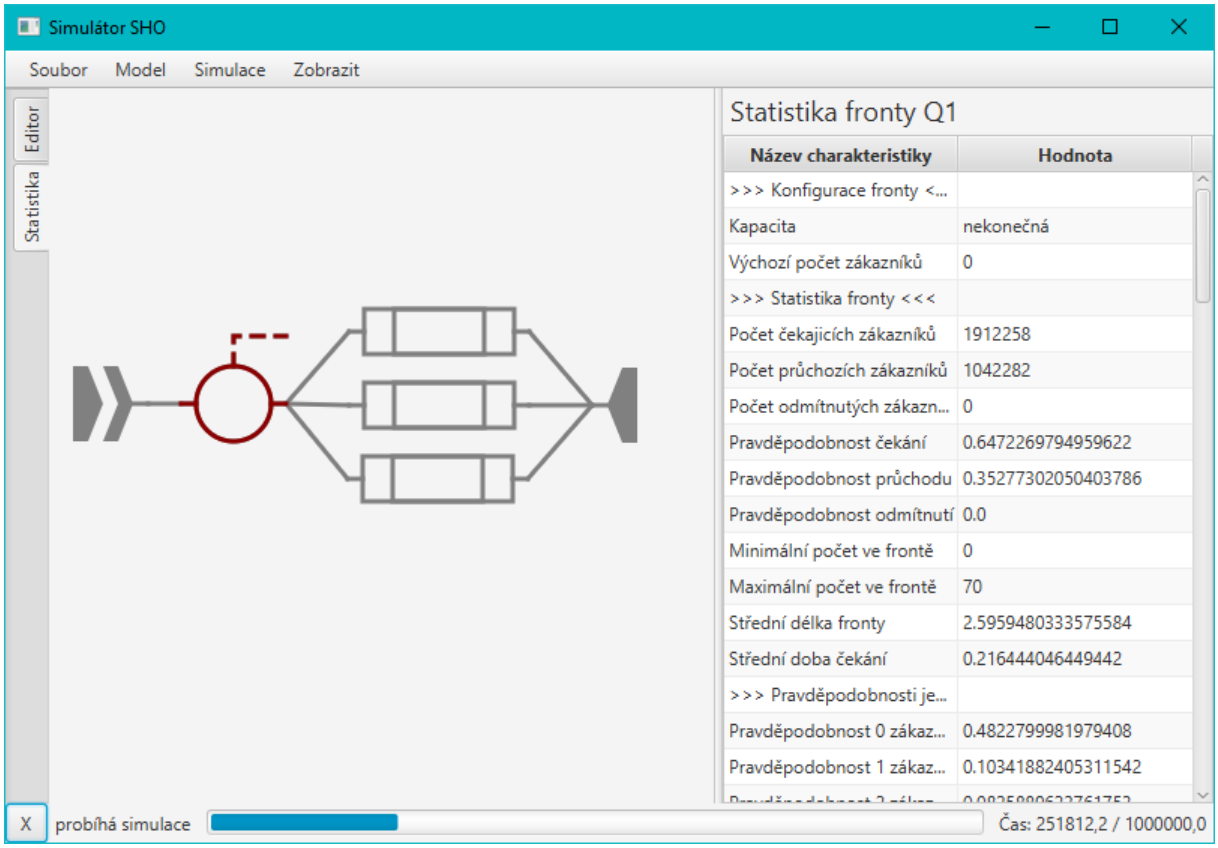
Pro řešení analytických modelů byly připraveny v tabulkovém procesoru LibreOffice Calc[6] patřičné vzorce a algoritmy. Řešení se zjednodušuje na pouhé zadání vstupních parametrů modelu a prakticky ihned jsou vypočítány všechny důležité charakteristiky SHO. Připravené vzorce v tabulkovém procesoru podporují všechny modely zmíněné v kapitole 2, tedy M|M|1, M|M|c, M|M|c|K, M|D|1 a M|G|1. Na obr. 18 je snímek obrazovky programu Calc při analytickém řešení modelu M|G|1.



Obrázek 18: Analytické řešení pomocí tabulkového procesoru. Zpracování autora.

[6]Program LibreOffice Calc představuje open-source alternativu ke komerčnímu programu Microsoft Excel.

Pro simulační řešení byl vyvinut vlastní software (viz kap. 3), který umožňuje ze základních komponent sestavit různé simulační modely. Kromě modelů zmíněných výše, simulátor umožňuje sestavení i dalších modelů, v této práci nepopisovaných — např. D|M|1, D|M|c, D|M|c|K, M|D|c, sériové M|M|c v tandemu a mnohé další. Na obr. 19 je snímek obrazovky simulačního softwaru při simulaci M|M|3 modelu.



Obrázek 19: Simulační řešení M|M|3 pomocí simulačního softwaru. Zpracování autora.

Pro každý individuální model byl vybrán ilustrační příklad, na kterém jsou demonstrovány oba přístupy, jak analytický, tak simulační. Příklady jsou smyšlené a nezakládají se na žádných faktických skutečnostech. Protože simulační řešení je založeno na generování náhodných čísel, každé provedení simulace má mírně odlišné výsledky. Z tohoto důvodu je zpravidla prováděno vždy několik experimentů se simulačním modelem. Pokud není uvedeno jinak, byl při simulacích nastaven celkový simulační čas simulace na 10^6 časových jednotek.

4.1 Model M|M|1

Příklad. Do ordinace praktického lékaře přichází průměrně 3 pacienti za hodinu. Bylo zjištěno, že příchod zákazníků je náhodná veličina odpovídající Poissonovu procesu. Ošetření pacienta u lékaře trvá průměrně 15 minut, přičemž doba ošetření má exponenciální rozložení pravděpodobnosti.

Tabulka 3: Analytické a simulační řešení modelu M|M|1

Veličina	**Analytické řešení**	**1. simulace**	**2. simulace**	**3. simulace**
$E[W_Q]$	0,7500	0,7463	0,7483	0,7475
$E[W_S]$	0,2500	0,2501	0,2498	0,2501
$E[W]$	1,000	0,9964	0,9944	0,9976
$E[N_Q]$	2,250	2,238	2,230	2,241
$E[N_S]$	0,7500	0,7500	0,7483	0,7500
$E[N]$	3,000	2,988	2,979	2,991
P_Q	0,7500	0,7499	0,7483	0,7499
P_S	0,2500	0,2501	0,2517	0,2501
p_0	0,2500	0,2500	0,2517	0,2500
p_1	0,1875	0,1875	0,1884	0,1876
p_2	0,1406	0,1406	0,1408	0,1406
p_3	0,1055	0,1056	0,1052	0,1052
p_4	0,07910	0,07934	0,07877	0,07898
p_5	0,05933	0,05949	0,05908	0,05935

Ze zadání ilustračního příkladu lze získat dva základní vstupní parametry pro model. Příchod zákazníků s parametrem $\lambda = 3$, intenzita obsluhy $\mu = 4$, přičemž za časovou jednotku je považována 1 hodina. Tabulka 3 obsahuje hodnoty analytického řešení společně s hodnotami získanými simulačními experimenty. Hodnoty jsou uváděny po zaokrouhlení na 4 platné číslice. Význam proměnných jednotlivých veličin je uveden v kapitole 1.3.2.

Z výsledků uvedených v tabulce 3 lze vyvodit několik závěrů. V první řadě je nutno konstatovat, že výsledky simulačního modelu jsou velmi podobné jejich teoretickým hodnotám. Průměrná doba pacienta stráveného u lékaře je 60 minut, z toho průměrně 45 minut čeká v čekárně a po dobu zbylých 15 minut je ošetřován. Příchozí pacient bude s pravděpodobností 0,25 přijat ihned k lékaři bez čekání. Vytížení lékaře je 75 %.

4.2 Model M|M|c

Příklad. Ve sledovaném intervalu přichází k pokladnám na vlakovém nádraží cestující v průměrném intervalu 11 s mezi příchody. Zákazník stráví u pokladny průměrně 28 s. Na nádraží jsou otevřeny celkem 3 pokladny. Je uvažováno, že intervaly mezi příchody zákazníků se řídí exponenciálním rozložením pravděpodobnosti, stejně tak i doby obsluhy.

Tabulka 4: Analytické a simulační řešení modelu M|M|3

Veličina	**Analytické řešení**	**1. simulace**	**2. simulace**	**3. simulace**
$E[W_Q]$	44,83	45,05	45,43	45,08
$E[W_S]$	28,00	27,99	28,02	28,00
$E[W]$	72,83	73,04	73,45	73,09
$E[N_Q]$	4,076	4,096	4,132	4,099
$E[N_S]$	2,545	2,546	2,548	2,546
$E[N]$	6,621	6,642	6,680	6,645
P_Q	0,7278	0,7281	0,7293	0,7282
P_S	0,2722	0,2719	0,2707	0,2718
p_0	0,04012	0,04012	0,03989	0,04009
p_1	0,1021	0,1020	0,1015	0,1019
p_2	0,1300	0,1298	0,1292	0,1298
p_3	0,1103	0,1102	0,1099	0,1102
p_4	0,09357	0,09347	0,09334	0,09344
p_5	0,07939	0,07919	0,07912	0,07924

Ze zádání příkladu vyplývá, že intenzita příchodu zákazníků $\lambda = \frac{1}{11}$ a intenzita obsluhy $\mu = \frac{1}{28}$. Počet linek obsluhy $c = 3$. Použitá základní časová jednotka je 1 sekunda. Tabulka 4 zobrazuje analytické řešení i výsledky simulačních experimentů. Cestující stráví čekáním ve frontě před pokladnou průměrně 45 s. Ve frontě čekají průměrně 4 cestující. Pravděpodobnost, že zákazník bude obsloužen ihned bez čekání ve frontě je zhruba 0,27.

Průměrné vytížení pokladen lze vypočítat jako:

$$\frac{E[N_S]}{c} = \frac{2,55}{3} = 0,85 \tag{115}$$

Veličiny $E[N_S]$ a $E[W_S]$ v tabulce 4 jsou získány jako průměr (veličina $E[W_S]$) nebo součet (veličina $E[N_S]$) ze všech linek obsluhy v modelu, protože simulátor vyhodnocuje každou linku obsluhy zvlášť. Také je nutno poznamenat, že THO předpokládá rovno-

měrně stejné vytížení všech linek obsluhy, zatímco při simulaci je vytížena nejvíce první a nejméně poslední linka obsluhy. Je to dáno implementací, kdy zákazník z fronty přichází vždy na první volnou linku, což může v realitě odpovídat např. situaci, kdy zákazníci volí vzdálenostně nejbližší volnou pokladnu, která poté může dosahovat vyššího vytížení, než pokladny ostatní.

4.3 Model M|M|c|K

Příklad. V bezprostřední blízkosti rušného městského silničního okruhu je provozována ruční myčka vozidel. Součástí myčky jsou mycí boxy, ve kterých si řidiči mohou umýt své osobní vozidlo. Průměrná doba strávená mytím vozidla je 8 minut. Bylo zjištěno, že v průběhu odpoledních hodin na myčku přijíždí v průměrů 13 vozidel za hodinu. Stavební uspořádání v bezprostřední blízkosti rušné silnice dovoluje, aby před mycími boxy čekala maximálně 4 vozidla. Více vozidel by již nebezpečně zasahovalo do přilehlé pozemní komunikace. Je předpokládáno, že příjezd vozidel tvoří Poissonův proces a doba obsluhy má exponenciální rozdělení pravděpodobnosti.

Tabulka 5: Analytické a simulační řešení modelu M|M|2|6

Veličina	**Analytické řešení**	**1. simulace**	**2. simulace**	**3. simulace**
$E[W_Q]$	0,09945	0,09951	0,09946	0,09949
$E[W_S]$	0,1333	0,1334	0,1333	0,1333
$E[W]$	0,2328	0,2329	0,2328	0,2328
$E[N_Q]$	1,164	1,164	1,164	1,164
$E[N_S]$	1,560	1,561	1,561	1,560
$E[N]$	2,724	2,725	2,725	2,724
P_Q	0,5783	0,5782	0,5784	0,5783
P_S	0,3219	0,3219	0,3217	0,3218
P_Z	0,09981	0,09998	0,09986	0,09991
p_0	0,1178	0,1176	0,1178	0,1178
p_1	0,2041	0,2044	0,2039	0,2041
p_2	0,1769	0,1770	0,1770	0,1768
p_3	0,1533	0,1531	0,1534	0,1533
p_4	0,1329	0,1327	0,1330	0,1328
p_5	0,1152	0,1153	0,1152	0,1153

Ze zadání vyplývá, že počet linek $c = 2$, maximální počet zákazníků v systému $K = 6$, intenzita příchodu zákazníků $\lambda = 13$ a intenzita obsluhy zákazníků $\mu = \frac{60}{8} = 7,5$. Základní časovou jednotkou je jedna hodina. Z výsledků obsažených v tabulce 5 je zřejmé, že průměrná doba čekání vozidla před mycím boxem je 6 minut (0,1 hodiny), v hrubém průměru je myčka obsazena 3 vozidly (2 v boxech a 1 čekající). Pravděpodobnost, že pro přijíždějící vozidlo již není místo před mycími boxy je 0,1. V 58 % případů vozidlo musí na umytí čekat, ve zbylých 32 % případech vozidlo vjíždí přímo do mycího boxu.

4.4 Model M|D|1

Příklad. Ve vodním ráji existuje pro návštěvníky několik zábavních atrakcí, mezi kterými je i dlouhý tobogán. Aby byla zajištěna bezpečnost návštěvníků, vstup do tobogánu je opatřen semaforem, který řídí rozestupy jednotlivých návštěvníků. Semafor udržuje rozestup dlouhý 15 s (tj. po dobu 15 s je zakázána jízda dalšího návštěvníka). K tobogánu přichází průměrně 180 zákazníků za hodinu. Příchod zákazníků tvoří Poissonův proces.

Tabulka 6: Analytické a simulační řešení modelu M|D|1

Veličina	**Analytické řešení**	**1. simulace**	**2. simulace**	**3. simulace**
$E[W_Q]$	0,3750	0,3741	0,3744	0,3732
$E[W_S]$	0,2500	0,2500	0,2500	0,2500
$E[W]$	0,6250	0,6241	0,6244	0,6232
$E[N_Q]$	1,125	1,122	1,123	1,119
$E[N_S]$	0,7500	0,7500	0,7501	0,7497
$E[N]$	1,875	1,872	1,873	1,869
P_Q	0,7500	0,7499	0,7499	0,7498
P_S	0,2500	0,2501	0,2501	0,2502
p_0	0,2500	0,2500	0,2499	0,2503
p_1	0,2793	0,2796	0,2793	0,2792
p_2	0,1942	0,1943	0,1944	0,1942
p_3	0,1167	0,1166	0,1170	0,1168
p_4	0,06764	0,06765	0,06791	0,06792
p_5	0,03902	0,03898	0,03885	0,03935

V příkladu přichází zákazníci do systému s intezitou $\lambda = 3$, zatímco intenzita obsluhy je $\mu = 4$. Časovou jednotkou je jedna minuta. Parametry modelu byly úmyslně zvoleny stejné jako pro model M|M|1, aby bylo možné zároveň i srovnat tyto modely mezi sebou.

Tabulka 6 obsahuje analytické i simulační řešení modelu. Průměrně ve frontě čeká jeden zákazník, přičemž doba čekání je průměrně 22 s. Využití tobogánu je 75 %. Pravděpodobnost čekání před tobogánem je 0,75.

Při porovnání modelu M|D|1 s modelem typu M|M|1 je zřejmé, že při stejných parametrech se v modelu M|D|1 tvoří fronta s nižší průměrnou délkou a zákazníci v průměru čekají kratší dobu.

V případě modelu M|D|1 se analytický model liší od simulačního. Analytický model pracuje s časovými okny konstantní doby obsluhy a pokud není na začátku tohoto časového okna ve frontě žádný zákazník, linka obsluhy zůstane po celou dobu neobsazena až do následujícího časového okna. V simulačním modelu se ovšem nečeká na žádné následující časové okno a obsluha zákazníka začíná ihned, pokud je linka obsluhy volná.

Pro daný příklad lze ilustrovat zmíněnou odlišnost následnovně. V případě analytického modelu se semafor zanedbatelně krátce rozsvítí zeleně a poté 15 s blokuje vstup do tobogánu, nehledě na to, zdali je v tobogánu nějaký návštěvník. Simulační model představuje semafor, který detekuje návštěvníka při vstupu do tobogánu a poté blokuje vstup 15 s. Pokud ovšem do tobogánu nevstoupí žádný zákazník (fronta je prázdná) zůstává na semaforu zelené světlo.

4.5 Model M|G|1

Příklad. Ve středně velké firmě zaměstnanci sdílí společnou tiskárnu, která je zapojena do počítačové sítě. Od různých zaměstnanců přichází v průběhu dne požadavky na tisk různých dokumentů. V průměru přichází 12 požadavků za hodinu. Příchody požadavků tvoří Poissonův proces. Není známo konkrétní pravděpodobnostní rozložení doby tisku jednoho dokumentu. Statisticky bylo zjištěno, že průměrná doba tisku dokumentu je 1 minuta, rozptyl této veličiny je 1.

Zkoumaný model M|G|1 má intenzitu příchodů $\lambda = 0,2$, doba obsluhy má střední hodnotu $E[S] = 1$ a rozptyl $\sigma^2 = 1$. Použitá časová jednotka je 1 minuta. Protože simulátor podporuje pouze exponenciální rozdělení pravděpodobnosti, bylo v simulaci využito právě ono. Pro použití exponenciálního rozdělení pravděpodobnosti musí být splněna následující

Tabulka 7: Analytické a simulační řešení modelu M|G|1

Veličina	**Analytické řešení**	**1. simulace**	**2. simulace**	**3. simulace**
$E[W_Q]$	0,2500	0,2511	0,2467	0,2494
$E[W_S]$	1,000	1,003	0,9988	1,001
$E[W]$	1,250	1,254	1,246	1,250
$E[N_Q]$	0,05000	0,05011	0,04911	0,04993
$E[N_S]$	0,2000	0,2000	0,1988	0,2004
$E[N]$	0,2500	0,2503	0,2479	0,2503

podmínka:

$$E[S]^2 = \sigma^2 \tag{116}$$

Z výsledků v tabulce 7 vyplývá, že střední doba čekání dokumentu v tiskové frontě je 15 s. Vytíženost tiskárny je 25 %.

ZÁVĚR

V práci jsou představeny různé modely systémů hromadné obsluhy, včetně metod jejich analytického řešení. Práce také obsahuje nezbytný úvod do teorie náhodných procesů a odvození matematických vzorců pro určení pravděpodobností a jiných charakteristik zkoumaných modelů systémů hromadné obsluhy.

Nemalé úsilí bylo věnováno vývoji simulačního softwaru, který by umožňoval simulaci různých modelů. Podařilo se vytvořit desktopovou aplikaci, která je napsána v programovacím jazyce Java a využívá grafické knihovny JavaFX. Zmiňovaná aplikace obsahuje modul simulačního jádra, díky kterému je možné provádět simulace programovou formou. Uplatnění tohoto modulu není jen pro grafickou nadstavbu, ale může najít uplatnění i u specifických a netypických simulací, u kterých je nutné získavat specifické simulační výstupy. Takové výstupy je možno získat implementací vlastních statistických tříd.

Zmiňovaná aplikace byla rozšířena o uživatelské rozhraní, čímž se simulační jádro zpřístupňuje všem uživatelům, kteří neumí nebo nechtějí simulační model popisovat jazykem Java. Grafická nadstavba využívá obsažených statistik pro jednotlivé komponenty, které by měly uspokojit většinu běžných potřeb uživatelů.

V rámci práce bylo provedeno několik simulací, přičemž jejich výsledky se přibližovaly analytickému řešení ekvivalentních modelů. U modelů, kde je známo analytické řešení, bych doporučil upřednostnit právě toto řešení. Analytické řešení poskytne přesnější hodnoty charakteristik. Pokud se pro potřebný model připraví potřebné vzorce (např. v tabulkovém procesoru), tak jsou výsledky získány rychleji než v případě simulace.

Výhoda simulace spočívá v použití pro systémy, jejichž analytické řešení není známo nebo je příliš komplikované. Pokud analytické řešení není známo, tak se vysloveně nabízí použít simulačního řešení. To ovšem musí být uzpůsobeno tak, aby simulační model systému nezanedbával podstatné vlastnosti reálného systému. V případě, že je analytické řešení příliš komplikované, může být zvolen simulační přístup, který sice nemusí dodat nejpřesnější výsledky, ale nevyžaduje komplikovaná matematická odvození.

Při řešení systémů hromadné obsluhy simulačním přístupem, může nastat problém s obtížností tvorby simulačního modelu. Tuto činnost se snaží co nejvíce zjednodušit aplikace vzniklá v této práci, popř. obdobné existující aplikace.

Vyvinutou aplikaci je možné do budoucna rozšířit o její další funkčnost implementací dalších pravděpodobnostních rozdělení, implementací specifických komponent apod. Možné využití je především jako jednoduše použitelný software pro zkoumání a počítačovou simulaci různě komplikovaných systémů hromadné obsluhy.

POUŽITÁ LITERATURA

[1] LINDA, Bohdan. *Stochastické metody operačního výzkumu.* Bratislava: Statis, 2004. ISBN 80-85659-33-6.

[2] ZIMOLA, Bedřich. *Operační výzkum.* Vyd. 4., nezm. Zlín: Univerzita Tomáše Bati, 2004. ISBN 80-7318-208-4.

[3] JABLONSKÝ, Josef. *Operační výzkum.* Vyd. 3. Praha: Vysoká škola ekonomická, 2001. ISBN 80-245-0162-7.

[4] PEŠKO, Štefan. *Teória hromadnej obsluhy: Prednášky pre AM* [online]. 2001. Žilina [cit. 2019-01-30]. Dostupné z: `http://frcatel.fri.uniza.sk/users/pesko/OA2/oa2m.pdf`

[5] GROSS, Donald. *Fundamentals of queueing theory.* 4th ed. Hoboken: John Wiley, c2008. Wiley series in probability and statistics. ISBN 978-0-471-79127-0.

[6] BACCELLI, François a Pierre BRÉMAUD. *Elements of queueing theory: Palm martingale calculus and stochastic recurrences.* 2nd ed. Berlin: Springer-Verlag, c2003. Applications of mathematics. Stochastic modelling and applied probability, 26. ISBN 3-540-66088-7.

[7] VORÁČOVÁ, Šárka. *Webskriptum - Teorie hromadné obsluhy* [online]. Praha [cit. 2019-02-10]. Dostupné z: `https://www.fd.cvut.cz/department/k611/PEDAGOG/K611THO.html`

[8] SIGMAN, Karl. *Lecture Notes on Stochastic Modeling I: Little's law* [online]. 2009. New York [cit. 2019-04-21]. Dostupné z: `http://www.columbia.edu/~ks20/stochastic-I/stochastic-I-LL.pdf`

[9] KOUTKOVÁ, Helena a Ivo MOLL. *Základy pravděpodobnosti.* Brno: Akademické nakladatelství CERM, 2008. Učební texty vysokých škol. ISBN 978-80-7204-574-7.

[10] MENDENHALL, William, Robert J BEAVER a Barbara M BEAVER. *Introduction to probability and statistics.* 12th ed. Belmont: Thomson Brooks/Cole, c2006. ISBN 0-534-41870-8.

[11] RÁBOVÁ, Zdena. *Modelování a simulace.* 3., přeprac. vyd. Brno: VUT, 1992. ISBN 80-214-0480-9.

[12] ROSS, Sheldon M. *Simulation.* Fifth edition. Amsterdam: Academic Press, 2013. ISBN 978-0-12-415825-2.

SEZNAM PŘÍLOH

Příloha A 82

PŘÍLOHA A

Příloha popisuje adresářovou strukturu a soubory na přiloženém CD.

```
/
|   dp_danhel.pdf
|   analyticke_reseni.ods
|-- simulacni_reseni
    |-- resene_modely
    |-- SimulatorSHO-linux64
    |-- SimulatorSHO-win64
    |-- zdrojovy_kod
```

Soubor `dp_danhel.pdf` obsahuje elektronickou verzi této práce ve formátu PDF (Portable Document Format).

Soubor `analyticke_reseni.ods` obsahuje vzorce a algoritmy umožňující analytické řešení modelů. Soubor je ve formátu OpenDocument Spreadsheet a byl vytvořen v programu LibreOffice Calc.

Adresář `simulacni_reseni` obsahuje vyvinutý software, který byl použit pro provádění simulací. Podadresář `resene_modely` obsahuje uložené modely, které lze načíst v simulační aplikaci a dále jsou zde obsaženy ukkázkové exporty statitistik z některých běhů simulace. Podadresář `SimulatorSHO-win64` obsahuje spustitelný kód aplikace, který je sestaven pro platformu Windows 64 bit. Podadresář `SimulatorSHO-linux64` obsahuje spustitelný kód aplikace, který je sestaven pro platformu Linux 64 bit. Podadresář `zdrojovy_kod` obsahuje zdrojové kódy programu v jazyce Java.