FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Výuková webová aplikace pro matematické modelování metodou MPM
Student:	Bc. Václav Dvořák
Vedoucí:	Ing. Petra Pavlíčková, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Webové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2021/2022

Pokyny pro vypracování

Návrh a implementace výukové webové aplikace pro matematické modelování na základě metody MPM (Metra Potential Method). Cílem této aplikace je vylepšení výuky matematického modelování a to přímo metody MPM.

1. Určete procesy, které bude aplikace podporovat a to především:
   1.a Vysvětlení metody na vzorových příkladech.
   1.b Vizuální editor pro práci s grafy.
   1.c Zobrazení vzorců na základě vytvořeného grafu.
   1.d Ukládání rozpracovaných příkladů
   2. Namapujte jednotlivé procesy na funkční požadavky aplikace.
3. Navrhněte grafické rozhraní aplikace.
4. Určete technologie, nad kterými bude vytvořena aplikace.
5. Připravte scénáře pro základní průchody aplikací.
6. Proveďte implementaci aplikace.
7. Otestujte aplikaci dle testovacích scénářů a opravte případné chyby.
8. Vytvořte uživatelskou a instalační příručku.
9. Proveďte zhodnocení a doporučení dalšího vývoje.

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 3. května 2020 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Výuková webová aplikace pro matematické modelování metodou MPM

Bc. Václav Dvořák

Katedra softwarového inženýrství
Vedoucí práce: Ing. Petra Pavlíčková, Ph.D.

5. května 2021

Poděkování

Tímto bych chtěl poděkovat paní Ing. Petře Pavlíčkové, Ph.D. za vedení této práce, její ochotu a čas strávený při konzultacích. Taktéž děkuji své rodině za podporu během celého studia.

Prohlášení

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

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

V Praze dne 5. května 2021 . . . . . .

Odkaz na tuto práci

Dvořák, Václav. *Výuková webová aplikace pro matematické modelování metodou MPM*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Diplomová práce se zabývá tvorbou webové aplikace pro výuku metody MPM. Metra Potential Method je jednou z metod sloužících k časové analýze projektu v rámci projektového řízení. Tato metoda je velice podobná metodě kritické cesty, kterou rozšiřuje o kladný a záporný potenciál jednotlivých vazeb mezi činnostmi. Díky těmto potenciálům vazeb umožňuje metoda pozměnit časový plán projektu bez složitých úprav topologie síťového diagramu. Vytvořená webová aplikace studentovi poskytuje veškerou teorii týkající se této metody včetně jejího rozšíření o různé typy vazeb. Aplikace dále obsahuje vizuální editor pro tvorbu síťových diagramů, na kterých je tato metoda simulována včetně popisu jejího výpočtu. Student si může vytvořit síťový diagram od první činnosti nebo nahrát několik vzorových příkladů, se kterými může dále pracovat.

Klíčová slova Projektové žízení, MPM, metra potential method, CPM, metoda kritické cesty, výuková aplikace, webová aplikace, síťové diagramy, React

Abstract

The diploma thesis deals with the creation of a web application for learning the MPM method. Metra Potential Method is one of the methods used for time analysis of a project within project management. This method is very similar to the critical path method, which it extends with the positive and negative potential of individual links between activities. Thanks to these potentials, the method allows to change the project schedule without complicated modifications to the network diagram topology. The created web application provides the student with all the theory concerning this method, including its extension by various types of links. The application also contains a visual editor for creating network diagrams on which this method is simulated, including a description of its calculation. The student can create a network diagram from the first activity or upload several sample examples, with which he can work further.

Keywords Project management, MPM, metra potential method, CPM, critial path method, educational application, web application, network diagrams, React

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Projektové řízení je nedílnou součástí každého úspěšného projektu. Jeho cílem je řízení neboli koordinace jednotlivých činností a zdrojů pro co nejefektivnější realizaci projektu – dosáhnutí požadovaného cíle. K časovému plánování projektů slouží několik metod založených na teorii grafů, mezi které patří mimo jiné metoda MPM (Metra Potential Method). Tato metoda na rozdíl od těch známějších, jako je metoda CPM (metoda kritické cesty), GERT nebo PERT, není tak známá. S metodou CPM má však hodně společného. Tématem této diplomové práce je vytvoření výukové aplikace pro výuku metody MPM, která studentům usnadní tuto metodu pochopit a to především kvůli obtížnému nalezení odborné literatury k této problematice.

Metodu vymyslel v roce 1958 francouzský výzkumník Bernard Roy pro firmu vyrábějící klikové hřídele [2]. Následně se metoda využívala při stavbě výletních lodí. Na začátku 60. let pak při vývoji leteckého programu Concorde a při stavbě první generace jaderných elektráren ve Francii. Termín projektové řízení v dnešní době většinou v lidech evokuje spojitost s IT prostředím, toto je však jasný příklad toho, že tato problematika vznikla v jiných odvětvích.

Teoretická část této práce nejdříve čtenáři vysvětlí pojem projektového řízení. Následně ho provede základními pojmy teorie grafů, které se používají při popisování jednotlivých metod síťové analýzy založených na síťových diagramech. Stěžejní částí teorie je popis jednotlivých metod síťové analýzy a jejich předchůdce – Ganttova diagramu. Základní metodou využívající síťové diagramy je metoda kritické cesty (CPM). Se znalostmi této metody lze jednodušeji pochopit princip metody MPM, která je motodě CPM velice podobná – umožňuje ohodnotit kromě uzlů i hrany grafu. Ohodnocením hran u této metody jsou takzvané potenciály vazeb. V teoretické části budou krátce popsány i další metody síťové analýzy, kterými jsou metody PERT a GERT.

Druhá část této práce se bude zabývat definováním jednotlivých procesů a funkčních požadavků na aplikaci. Z těchto informací budou odvozeny nefunkční požadavky na aplikaci tak, aby co nejvíce vyhovovaly bu-

doucímu použití. Na základě vydefinování funkčních požadavků budou odvozeny případy užití, které budou znázorněny UML diagramem a následně detailněji vysvětleny pomocí scénářů popisující základní průchody aplikací. Mezi tyto průchody lze zařadit například zobrazení teorie metody MPM, simulace metody ve vizuálním editoru nebo ukládání a načítání rozpracovaných příkladů. Tato část diplomové práce bude zakončena návrhem uživatelského rozhraní, během kterého budou vytvořeny drátěné modely jako podklad pro následnou implementaci.

Ve zbylých kapitolách této práce se budu zabývat samotnou implementací výukové aplikace, jejím testováním a případným návrhem dalšího vývoje. Pro testování aplikace budou zhotoveny testovací scénáře, dle kterých bude aplikace uživatelsky otestována. Na základě tohoto testování budou opraveny případné chyby.

Kapitola 1

Cíl práce

Cílem této diplomové práce je vytvoření webové výukové aplikace pro matematické modelování metodou MPM, která se využívá v projektovém řízení pro časovou analýzu projektu. Aplikace by měla zjednodušit výuku této metody studentům bakalářského studia fakulty informačních technologií na ČVUT. Měla by studentům metodu vysvětlit a umožnit jim vyzkoušet si ji na vzorových příkladech ve vizuálním editoru. Studenti si dále budou moci vytvářet příklady vlastní a ukládat rozpracované řešení. Hlavní částí vizuálního editoru bude práce se síťovým diagramem znázorňující jednotlivé dílčí činnosti projektu a vazby mezi nimi. Na základě vytvořeného grafu bude zobrazen celý postup směřující k výsledku metody. Cíle této práce lze shrnout do následujících kroků:

- shrnout teoretické informace k základům teorie grafů, na kterých jsou založeny metody pro časovou analýzu projektů
- shrnout teoretické informace k MPM metodě
- určit procesy, které bude aplikace podporovat
- vytvořit případy užití
- navrhnout grafické rozhraní aplikace
- určit technologie, které budou pro vývoj použity
- implementovat webovou aplikaci
- vytvořit vzorové příklady a vložit je do aplikace
- otestovat aplikaci dle testovacích scénářů
- vytvořit dokumentaci
- zhodnotit aplikaci a doporučit další vývoj

Kapitola 2

Teorie

2.1 Projektové řízení

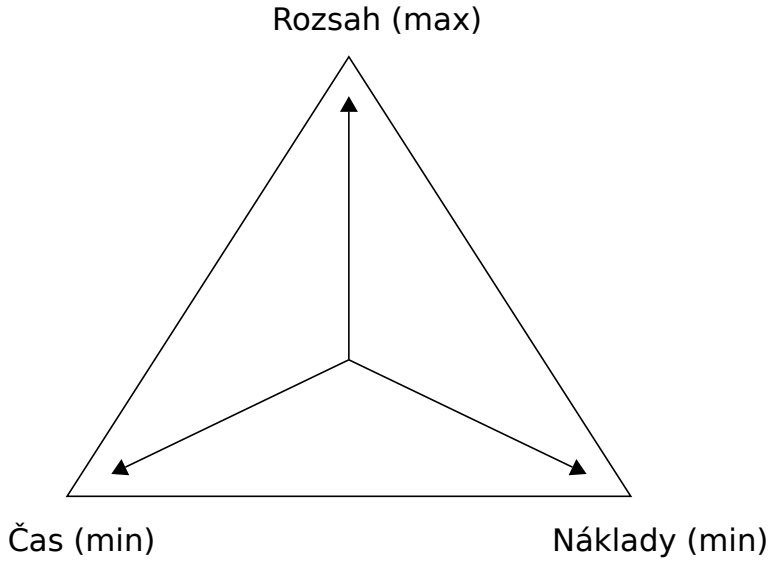
Projekt je *„časově omezená pracovní činnost, jejímž cílem je vytvoření jedinečného produktu, služby, nebo dosažení jiného výsledku.“* [3]. Od klasické operační nebo provozní činnosti se liší tím, že končí po dosažení stanovených cílů. V informačních technologiích lze jako projekt považovat jakoukoliv zakázku, která je dodávaná klientovi. Každý takový projekt je omezen třemi kritérii – trojimperativ projektu [4]:

- **Rozsah (výsledky):** Specifikace jednotlivých funkcionalit výsledného produktu nebo služby. Jaké všechny požadavky klade na výsledný produkt zákazník. Kvalitní specifikace je základním stavebním kamenem pro úspěšný projekt. Je užitečná jak pro zadavatele, tak zpracovatele. Obě strany si jasně definují všechny cíle a vyhnou se tak případným nedorozuměním. Zadavatel nemůže po dokončení vznést námitky, že očekával nějakou chybějící funkcionalitu a naopak zpracovatel při kvalitní specifikaci nemůže zadavatele v případě sporu podvést.

- **Čas:** Kolik času bude potřeba ke splnění všech cílů a za jak dlouhou dobu musí být splněny. Díky kvalitní specifikaci a časové dotaci na projekt dokážeme určit, kolik zdrojů musíme pro projekt uvolnit. Tedy například, kolik vývojářů musí na projektu pracovat, aby se stihl dokončit v požadovaném termínu.

- **Náklady (zdroje):** Jaké budou výsledné náklady na dokončení projektu? Na základě rozsahu projektu a potřebného času k jeho dokončení můžeme odhadnout potřebné zdroje k dokončení a výsledný rozpočet.

Projektové řízení je *„uplatnění veškerých poznatků, dovedností, nástrojů a technik na aktivity (činnosti) projektu takovým způsobem, aby byly splněny*



Obrázek 2.1: Trojimperativ projektu

požadavky na projekt“ [3]. Projektový manažer se snaží v rámci řízení projektů splnit výše zmíněný trojimerativ. Jeho práce spočívá v následujících pěti krocích [5]:

- **Definování:** *„Definování projektových cílů.“*
- **Plánování:** *„Naplánování, jak vy a váš tým splníte podmínky – trojimperativ, tj. specifikace provedení, časový plán a finanční rozpočet (plán závisí na poměru lidských a materiálních zdrojů, které mají být použity.)“*
- **Vedení:** *„Uplatnění manažerského stylu řízení lidských zdrojů, podřízených a jiných (včetně subdodavatelů), který je povede k tomu, že svou práci budou vykonávat efektivně a včas.“*
- **Sledování (monitorování):** *„Kontrola stavu a postupu projektových prací, abyste včas zjistili odchylky od plánu a mohli jste rychle přistoupit k jejich korekci. (To často vede k úpravám plánu, které si mohou vynutit i změnu cíle [definice], a v důsledku toho i potřebu změny zdrojů.)“*
- **Ukončení:** *„Ověření, že hotový úkol odpovídá aktuální definici toho, co se mělo udělat, a uzavření všech nedokončených prací, např. dokumentace.“*

K řízení času se v projektovém řízení využívají Ganttovy diagramy, síťové diagramy a metody pracující s nimi, jako jsou metody CPM, PERT, GERT či Metra potential method, která je tématem této diplomové práce.

2.2 Teorie grafů

Metody časového plánování založené na síťové analýze využívají síťové diagramy (grafy). Z tohoto důvodu je zapotřebí znát základy teorie grafů. Teorie grafů je samostatný obor diskrétní matematiky, jehož původ sahá až do 18. století. Jejím zakladatelem je švýcarský matematik a fyzik Leonhard Paul Euler, který řešil slavný problém sedmi mostů města Královce [6].

Graf je matematická struktura, která je definována jako uspořádaná dvojice neprázdné konečné množiny vrcholů V, neboli vertices, a konečné množiny hran E, neboli edges. Každá hrana z množiny E je určena dvouprvkovou množinou vrcholů z množiny V. Formálně graf zapisujeme jako:

$$G = (V, E)$$

$$E \subseteq \{\{u, v\} | u, v \in V, u \neq v\}$$

Pro práci s metodami časového plánování založených na síťové analýze je potřeba znát následující definice [7]:

Orientovaný graf Orientovaný graf je takový graf, jehož hrany jsou reprezentovány jako uspořádaná dvojice, díky čemuž je identifikovatelný směr hrany. Naopak hrany neorientovaného grafu jsou dvouprvkové množiny. Výrazy (x, y) a (y, x) označují v orientovaném grafu dvě rozdílné hrany, zatímco v neorientovaném grafu jsou hrany $\{x, y\}$ a $\{y, x\}$ identické. V orientovaném grafu jsou hrany znázorněny šipkami.

Stupeň uzlu Stupeň uzlu $d(v)$ je počet hran, které s uzlem $v \subseteq V(G)$ incidují (jsou spojeny s uzlem). U orientovaných grafů se dále stupěň uzlu dělí na:

- Stupeň vstupní $d^+(v)$: počet hran, které jsou orientovány směrem do uzlu
- Stupeň výstupní $d^-(v)$: počet hran, které jsou orientovány směrem z uzlu

Celkový stupeň uzlu v orientovaném grafu je součet stupně vstupního a výstupního:

$$d(v) = d^+(v) + d^-(v)$$

Součet stupňů všech uzlů v grafu je roven dvojnásobku počtu hran (handshaking lemma). Důsledkem toho je, že v každém grafu se nachází sudý počet uzlů lichého stupně.

$$\sum_{\forall v \in V} d(v) = 2|E|$$

Podgraf Podgraf si lze představit jako podmnožinu nějakého grafu. Je to graf, který vznikne odebráním některých vrcholů a hran z původního grafu. Graf $H = (V_H, E_H)$ je podgraf grafu $G = (V_G, E_G)$, jestliže platí následující podmínky:

- $V_H \subseteq V_G$
- $E_H \subseteq E_G$
- Hrany grafu H mají oba vrcholy v grafu H

Kružnice Kružnice je graf, který se skládá z jediného cyklu – z uzavřené posloupnosti propojených vrcholů hranami. V závislosti na tom, jestli je graf orientovaný, rozlišujeme kružnici na orientovanou a neorientovanou. Kružnici značíme jako $C_n = (V, E)$, kde $V = \{v_1, ..., v_n\}$ a $E = \{e_1, ..., e_n\}$ a platí:

- pro orientovaný graf:
  - $e_i = (v_i, v_{i+1}), i = 1, ..., n-1$ a $e_n = (v_n, v_1)$
  - $d^+(v) = d^-(v) = 1$ – každý vrchol $v \in V(C)$ má výstupní i vstupní stupeň roven 1
- pro neorientovaný graf:
  - $e_i = \{v_i, v_{i+1}\}, i = 1, ..., n-1$ a $e_n = \{v_n, v_1\}$
  - $d(v) = 2$ – každý vrchol $v \in V(C)$ má stupeň roven 2

Acyklický graf Acyklický graf je graf, který neobsahuje žádný cyklus. Neboli je to graf, který neobsahuje jako podgraf kružnici. V opačném případě je graf cyklický.

Vážený graf Nebo také ohodnocený graf, je takový graf, jehož všechny hrany $e \subseteq E(G)$ jsou ohodnoceny reálným číslem pomocí funkce $\omega = E \to \mathbf{R}$. Takové číslo nazýváme váhou hrany. Kladně ohodnocený graf G má takové ohodnocení ω, že pro každou hranu $e \subseteq E(G)$ je její váha $\omega(e)$ kladná.

Cesta a její délka Termínem cesta se v teorii grafů označuje posloupnost $P = (v_0, e_1, v_1, ..., e_n, v_n)$ pro kterou platí $e_i = \{v_{i-1}, v_i\}$, nebo v případě orientovaného grafu $e_i = (v_{i-1}, v_i)$. Navíc platí $v_i \neq v_j$, $i \neq j$ – žádné dva vrcholy ani hrany se v posloupnosti neopakují.

Délka cesty je definována různě. Může to být počet vrcholů nebo počet hran posloupnosti. V případě váženého (ohodnoceného) grafu je délka cesty definována jako součet vah všech hran posloupnosti.

Shled Shled v grafu $G = (V, E)$ je posloupnost vrcholů $v \in E(V)$ taková, že mezi každými dvěma po sobě jdoucími vrcholy v_i, v_{i+1} existuje hrana $e = \{v_i, v_{i+1}\}$. Na rozdíl od cesty se mohou vrcholy opakovat. V případě orientovaného shledu v orientovaném grafu musí existovat orientovaná hrana $e = (v_i, v_{i+1})$.

Souvislý graf Souvislý graf je takový graf, v němž pro každé dva vrcholy $x, y \in E(V)$ existuje shled z x do y. Pro orientované grafy se navíc rozlišuje mezi slabou a silnou souvislostí.

Síťový diagram Neboli síťový graf je orientovaný graf, který je nejvhodnější technikou pro znázornění činností a jejich závislostí v projektovém řízení. Využívá se téměř ve všech metodách pro časové plánování projektů. Při vytváření síťových grafů musí být dodržena následující pravidla:

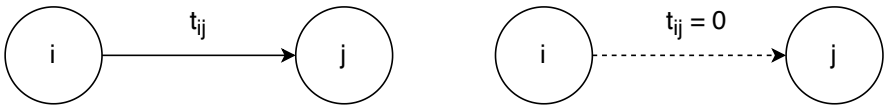
- Síťový diagram je souvislý.
- Síťový diagram může mít pouze jeden počáteční uzel (začátek projektu).
- Síťový diagram může mít pouze jeden koncový uzel (konec projektu).
- Do každého uzlu, kromě počátečního, musí vést minimálně jedna hrana – vstupní stupeň $d^+(v) > 0$.
- Z každého uzlu, kromě koncového, musí vést minimálně jedna hrana – výstupní stupeň $d^-(v) > 0$.
- Mezi jakýmikoliv dvěma uzly může vést maximálně jedna hrana.
- Síťový diagram je acyklický – neobsahuje kružnici jako podgraf.

2.3 Přehled metod časového plánování

Každý projekt by se měl rozložit do hierarchické struktury činností (WBS – Work breakdown structure [8]). Z této struktury jsou jednoduše identifikovatelné jednotlivé činnosti, čas potřebný pro jejich dokončení a logické vazby mezi nimi, ze kterých lze snadno vytvořit úsečkový nebo síťový diagram. Síťové diagramy dále dělíme na dvě základní skupiny podle zápisu jednotlivých činností [5]:

AOA Activity on arrow, česky činnost na hraně (šipce), využívá hrany grafu pro reprezentaci jednotlivých činností. Někdy se tento typ síťový diagramů označuje rovněž jako ADM (arrow diagram method) nebo hranově ohodnocený graf. Tato metoda se vyznačuje následujícími vlastnostmi:

- Uzly v grafu se využívají pro reprezentaci událostí (začátku či konce činnosti).
- Hrany se využívají pro reprezentaci činností. Rozeznáváme dva druhy činností (obrázek 2.2):
  - **reálné:** Skutečné činnosti, které se realizují, stojí nějaké zdroje a trvají nějaký čas. V grafu se znázorňují plnou čarou.
  - **fiktivní:** Vyjadřují pouze závislost (nějaké omezení) mezi událostmi. Tyto činnosti mají nulovou dobu trvání, nestojí žádné zdroje a v grafu se obvykle znázorňují přerušovanou čárou.
- Ohodnocení hrany znázorňuje dobu trvání činnosti.
- Jediným typem vazeb mezi uzly je tzv. *„finish to start“* (FS) vazba. Tedy činnost probíhá od konce nějaké události do začátku následující události.



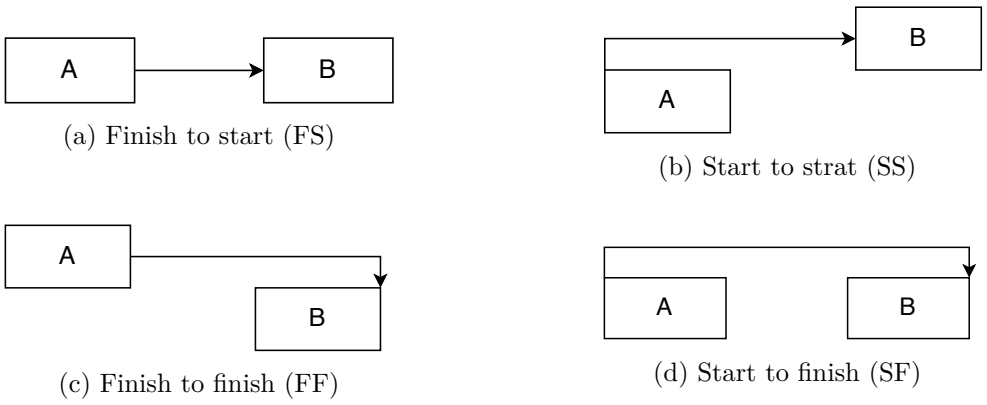
(a) Reálná činnost (b) Fiktivní činnost

Obrázek 2.2: Činnosti v síťovém diagramu typu AOA

AON Activity on node, česky činnost v uzlu, využívá uzly grafu pro reprezentaci jednotlivých činností. Hrany znázorňují závislosti mezi činnostmi. Někdy se tento typ síťový diagramů označuje rovněž jako PDM (precedence diagram method) nebo uzlově ohodnocený graf. Na rozdíl od AOA umožňuje čtyři typy vazeb mezi činnostmi A a B (obrázek 2.3):

- **finish to start** (FS): Činnost B může začít po dokončení činnosti A.
- **start to start** (SS): Činnost B nemůže být zahájena před začátkem činnosti A.
- **finish to finish** (FF): Činnost B nemůže být dokončena před dokončením činnosti A.
- **start to finish** (SF): Činnost B nemůže být dokončena před zahájením činnosti A.

Oba typy síťových diagramů lze mezi sebou v případě metody kritické cesty převádět. V dnešní době je častěji používaný typ AON, který je více intuitivní.



(a) Finish to start (FS)

(b) Start to strat (SS)

(c) Finish to finish (FF)

(d) Start to finish (SF)

Obrázek 2.3: Typy vazeb v síťovém diagramu typu AON

V uzlově definovaném síťovém diagramu je možné ohodnotit i vazby určitými hodnotami. Tento přístup se využívá například v metodě MPM, kde se hrany ohodnocují kladným a záporným potenciálem vazby.

2.3.1 Ganttův diagram

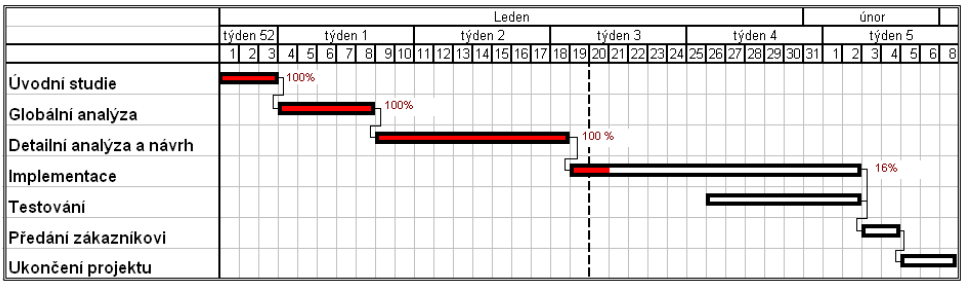
Ganttovy diagramy, také nazývány jako úsečkové diagramy, jsou pruhové diagramy pojmenované podle průmyslového inženýra H. L. Gantta [9]. Gantt však nebyl jeho vynálezcem, nýbrž je považován za průkopníka v jeho používání během první světové války. První dochovaný Ganttův diagram byl vytvořen v roce 1896 Karolem Adamieckim, který ho nazýval harmonogramem. Adamicki svůj diagram publikoval až po tom, co ho proslavil Gantt, proto je pojmenován právě po něm.

Ganttovy diagramy se staly standardním formátem pro grafické znázornění časových plánů projektů. Na horizontální ose diagramu je znázorněno časové období projektu rozdělené na stejně velké celky, z pravidla na dny, týdny nebo měsíce. Na vertikální ose diagramu jsou vyjmenovány jednotlivé činnosti projektu. Na ploše diagramu jsou formou pruhů znázorněné jednotlivé činnosti vzhledem k časové ose. Začátek pruhu indikuje začátek činnosti a konec pruhu indikuje její konec. Jednotlivé pruhy mají vždy stejnou výšku. Nelze tedy v Ganttovu diagramu vyjádřit náročnost projektu na zdroje.

V základní verzi Ganttova diagramu nelze definovat závislosti mezi jednotlivými činnostmi. V rozšířené verzi Ganttova diagramu se závislosti mezi činnostmi znázorňují pomocí šipek nebo čar. V digitální podobě diagramu je dále často znázorňován aktuální den formou svislé linky a procentuální dokončení dané činnosti (obrázek 2.4).

Mezi další rozšíření patří například přidávání milníků nebo znázornění souhrnných úkolů, neboli seskupení několika činností do sebe. Milník je významná událost nebo cíl v průběhu projektu. Bývá to konec nějaké etapy, částečný výstup předávaný zákazníkovy apod.

Ganttovy diagramy jsou snadno srozumitelné pro menší projekty. U projektů obsahujících mnoho činností mohou být již nepraktické a nepřehledné. Hlavním důvodem je, že sdělují relativně málo informací na jednotku plochy.



Obrázek 2.4: Příklad jednoduchého Ganttova diagramu s vyznačenými závislostmi činností a mírou jejich dokončení [1]

2.3.2 Metoda CPM

Metoda kritické cesty, neboli critical path method (CPM), je matematická metoda sloužící k plánování průběhu projektu a odhadnutí celkové doby trvání projektu [4]. Lze ji využít jak na síťovém diagramu typu AOA, tak na AON. V rámci této práce vysvětlím metodu kritické cesty na síťovém diagramu typu AON. Tato metoda byla vyvinuta v 50. letech 20. století Morganem R. Walkerem ze společnosti DuPont a Jamesem E. Kelleyem Jr. ze společnosti Remingto Rand. V současné době se používá ve všech druzích projektů od stavebnictví až po vývoj softwaru. Jedná se o deterministickou metodu, ve které se každý uzel (činnost) realizuje v momentě, kdy skončí realizace všech činností, na kterých je závislá (vstupní hrany). Následně dokončení činnosti zahájí realizaci všech činností, které na ni závisí (výstupní hrany).

Vstupem do této metody je seznam činností, dob jejich trvání a závislostí mezi nimi. Z těchto hodnot je sestaven síťový diagram. Následně je proveden tzv. výpočet vpřed, kdy algoritmus postupuje od počátečního uzlu (začátku projektu) a počítá k jednotlivým činnostem termíny jejich nejdříve možného začátku a konce. V druhé fázi výpočtu, tzv. výpočtu vzad, algoritmus postupuje od koncového uzlu (konce projektu) a k jednotlivým činnostem počítá termíny jejich nejpozději přípustného začátku a konce. Na základě těchto hodnot jsou určeny časové rezervy jednotlivých činností a kritická cesta celého síťového grafu. Ta je definována jako cesta grafem, na které nejsou žádné časové rezervy. Jakékoliv časové zpoždění jedné z činností v kritické cestě by znamenalo zpoždění celého projektu. Různé literatury se liší ve značení jednotlivých veličin, v rámci této práce bude použito následující značení:

- t_i - dobra trvání činnosti i

- ES_i - termín nejdříve možného začátku činnosti (Early Start)
- EF_i - termín nejdříve možného konce činnosti (Early Finish)
- LS_i - termín nejpozději přípustného začátku činnosti (Late Start)
- LF_i - termín nejpozději přípustného konce činnosti (Late Finish)
- F_i - interferenční časová rezerva činnosti (Float)

Obrázek 2.5: Reprezentace uzlu v metodě CPM

Průběh celého výpočtu metody CPM popíši na následujícím vzorovém příkladu. Pro příklad jsem definoval činnosti, jejich dobu trvání a závislosti mezi nimi podle tabulky 2.1. Na obrázku 2.6, na konci této kapitoly, je graficky znázorněn celý průběh výpočtu.

Tabulka 2.1: Příklad činností, jejich doby trvání a závislostí pro metodu CPM

Název činnosti	Doba trvání	Závislost na
A	6	-
B	4	-
C	3	A
D	6	B
E	3	B
F	2	C, D
G	3	E

Výpočet vpřed U výpočtu vpřed postupujeme od počátečního fiktivního uzlu – začátek projektu. Tento uzel pouze znázorňuje počátek projektu a slouží ke splnění podmínek síťového diagramu, který může mít pouze jeden počáteční uzel. U všech uzlů, do kterých vstupuje hrana pouze z počátečního fiktivního uzlu, je nejdříve možný začátek činnosti $ES_i = 0$ a nejdříve možný konec $EF_i = t_i$. Činnosti (uzly), na kterých je činnost i závislá, označme jako j. Pro všechny ostatní uzly dále platí:

$$ES_i = \max(ES_j + t_j)$$

$$EF_i = ES_i + t_i$$

Výpočet vzad Výpočet vzad je podobný výpočtu vpřed. Na rozdíl od něho postupujeme od konce projektu – fiktivního koncového uzlu. U všech uzlů, ze kterých vede hrana do fiktivního koncového uzlu, je nejpozději přístupný konec činnosti maximální hodnota nejdříve možného konce činnosti z této množiny uzlů. Nepozději přípustný začátek těchto činností je $LS_i = LF_i - t_i$. Činnosti (uzly), do kterých vede hrana z uzlu i označíme jako j. Pro všechny ostatní uzly dále platí:

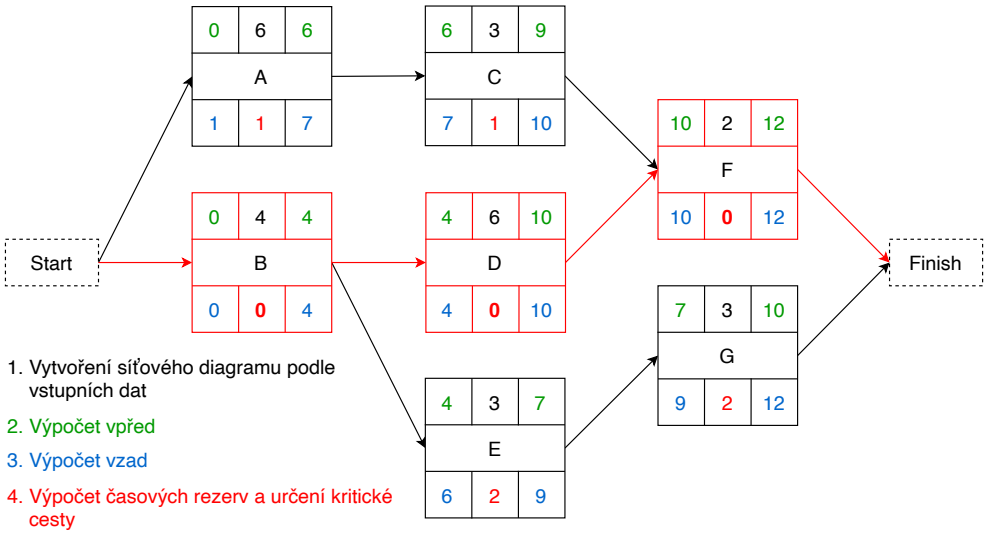
$$LF_i = \min(LF_j - t_j)$$

Výpočet časových rezerv a určení kritické cesty Na základě vypočtených termínů jednotlivých činností je možné vypočítat jejich interferenční časovou rezervu:

$$F_i = LS_i - ES_i = LF_i - EF_i = LF_i - ES_i - t_i$$

Všechny uzly s nulovou interferenční časovou rezervou jsou kritické činnosti a jsou součástí kritické cesty, která je výsledkem této metody. Jakékoliv zpoždění jedné z činností na kritické cestě by znamenalo zpoždění celého projektu.

Při výpočtu kritické cesty pomocí interferenční časové rezervy se může stát, že je kritických cest v síťovém diagramu více a nedá se jednoznačně určit, která je ta správná. K důkladnějšímu rozboru slouží další druhy časových rezerv jako celková časová rezerva, volná časová rezerva nebo nezávislá časová rezerva.



Obrázek 2.6: Síťový diagram – Ukázka výpočtu pomocí metody CPM

2.3.3 Metoda MPM

Tuto metodu vymyslel v roce 1958 francouzský výzkumník Bernard Roy pro firmu vyrábějící klikové hřídele [2]. Následně se metoda využívala při stavbě výletních lodí. Na začátku 60. let pak při vývoji leteckého programu Concorde a při stavbě první generace jaderných elektráren ve Fancii.

MPM metoda je velmi podobná metodě kritické cesty popsané v kapitole 2.3.2. Pro práci s touto metodou je vždy využito síťových diagramů typu AON. Ohodnocení uzlů znázorňuje dobu trvání činnosti. Na rozdíl od metody CPM tato metoda umožňuje i ohodnocení hran, které vychází ze dvou typů vztahů mezi činnostmi [10]. Těmto vztahům se říká potenciály vazby. Rozlišují se dva typy potenciálů vazeb mezi uzly i a j:

- **kladný potenciál vazby** $a_{i,j}$: určuje minimální časový odstup dvou po sobě jdoucích činností i a j
- **záporný potenciál vazby** $b_{i,j}$: určuje maximální časový odstup dvou po sobě jdoucích činností i a j

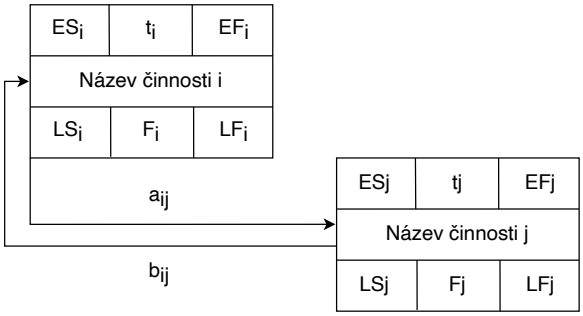
Díky potenciálům vazeb tato metoda, na rozdíl od metody CPM, umožňuje pozměnit časový plán projektu bez složitých úprav topologie síťového diagramu.

Záporný potenciál vazby $b_{i,j}$ se standardně uvádí pomocí hrany orientované proti směru hrany s kladným potenciálem. Pro zjednodušení lze tuto opačnou hranu vynechat a uvést oba potenciály na jednu hranu (obrázek 2.7c). Metoda MPM, na rozdíl od metody CPM, nemusí mít vždy řešení. Aby tato metoda měla řešení, musí být splněny následující podmínky:

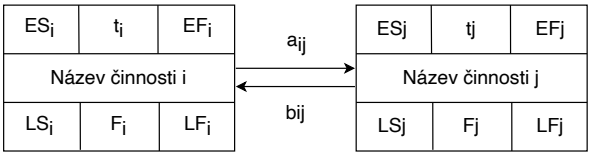
- na všech kružnicích grafu musí být součet absolutních hodnot záporných potenciálů větší nebo roven součtu kladných potenciálů
- u každé dvojce hran mezi uzly i a j musí být kladný potenciál menší nebo roven absolutní hodnotě záporného potenciálu – $a_{i,j} \leq |b_{i,j}|$
- pokud není hrana ohodnocena kladným potenciálem, tak $a_{i,j} = 0$
- pokud není hrana ohodnocena záporným potenciálem, tak $b_{i,j} = -\infty$

Metoda v původní variantě pracuje pouze s vazbami typu *„start to start“*. Rozšíření této metody bude probráno v následující kapitole 2.3.3.1. Až na potenciály vazeb zůstává značení stejné jako u metody CPM. Vzhledem k typu vazby by se měl síťový diagram zakreslovat podle obrázku 2.7a. Pokud uvažujeme pouze základní variantu s typem vazeb *„start to start“*, lze síťový diagram zjednodušit podle obrázku 2.7b.

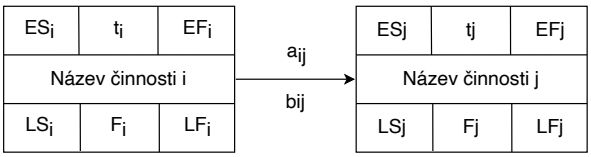
Pro kladný potenciál vazby $a_{i,j}$ musí platit pravidlo $ES_j \geq ES_i + a_{i,j}$. Definováním hodnoty $a_{i,j}$ vzniká jeden ze tří případů:



(a) Značení MPM vazby typu start to start



(b) Zjednodušené značení MPM vazby typu start to start při využití jednoho typu vazby



(c) Využití jedné hrany pro kladný i záporný potenciál vazby

Obrázek 2.7: Reprezentace uzlů a vazby v metodě MPM

- $a_{i,j} > 0$: událost j může začít nejdříve za $a_{i,j}$ časových jednotek po začátku události i
- $a_{i,j} = 0$: událost j může začít nejdříve v čas začátku událost j
- $a_{i,j} < 0$: událost j může začít nejdříve $a_{i,j}$ časových jednotek před začátkem události i

Pro záporný potenciál vazby $b_{i,j}$ musí platit pravidlo $ES_j \leq ES_i - b_{i,j}$. Definováním hodnoty $b_{i,j}$ vzniká jeden ze tří případů:

- $b_{i,j} > 0$: událost j může začít nejpozději $a_{i,j}$ časových jednotek před začátkem události i
- $b_{i,j} = 0$: událost j musí začít nejpozději v čas začátku události i
- $b_{i,j} < 0$: událost j může začít nejpozději za $a_{i,j}$ časových jednotek po začátku události i

Stejně jako u metody CPM, tak i v této metodě probíhají výpočty vpřed a vzad. Na rozdíl od metody CPM se v této metodě nepočítá jen s dobou trvání činnosti t_i, ale i s kladným a záporným potenciálem vazby.

Výpočet vpřed Pro všechny počáteční činnosti v síťovém diagramu, stejně jako v CPM, platí $ES_i = 0$ a $EF_i = t_i$. Pro všechny ostatní činnosti se určí nejdříve možný konec činnosti EF_i a z něho se dopočítá nejdříve možný začátek činnosti $ES_i = EF_i - t_i$. Označme všechny předcházející činnosti, které mají orientovanou hranu do činnosti j jako i. Poté můžeme vypočíst nejdříve možný konec j této činnosti jako:

$$EF_j = max(ES_i + a_{i,j} + t_j)$$

Na závěr je potřeba ověřit, zda hodnoty ES_i a ES_j splňují podmínku záporného potenciálu vazby $ES_j - ES_i \leq -b_{i,j}$. Pokud tuto podmínku záporné potenciály nesplňují, tak metoda nemá řešení a výpočet je pozastaven. V takovém případě je potřeba upravit buď potenciály vazeb nebo dobu trvání činností.

Výpočet vzad Při výpočtu vzad se stejně jako v metodě CPM postupuje od koncových činností i, pro které platí $LF_i = max(EF_i)$, a $LS_i = LF_i - t_i$. Pro každou další činnost i označme všechny činnosti, do kterých vede orientovaná hrana z činnosti i jako j. Poté můžeme vypočíst nepozději možný začátek činnost i jako:

$$LS_i = min(LS_j - a_{i,j})$$

Termín nejpozději možného konce činnosti i vypočteme jako $LF_i = LS_i + t_i$. Stejně tak jako u výpočtu vpřed je potřeba pro všechny vazby ověřit podmínku pro záporný potenciál $LF_j - LF_i \leq -b_{i,j}$. Pokud tato podmínka není splněna, tak je potřeba upravit hodnoty jednotlivých činností nebo vazeb jako při výpočtu vpřed.

Výpočet časových rezerv a určení kritické cesty V této finální fázi je opět MPM metoda velice podobná metodě CPM. K jednotlivým činnostem vypočteme jejich časovou rezervu pomocí vzorce:

$$F_i = LS_i - ES_i = LF_i - EF_i = LF_i - ES_i - t_i$$

Činnosti s nulovou časovou rezervou jsou činnosti kritické. Posloupnost těchto kritických činností tvoří kritickou cestu. Oproti metodě CPM je potřeba brát v potaz kladné potenciály vazeb. Do kritické cesty jsou zahrnuty pouze vazby, které tvoří maximální hodnotu nejdříve možného konce kritické činnosti.

2.3.3.1 Možnosti rozšíření MPM metody

V základní verzi MPM metody jsou použity pouze vazby typu *„start to start“*. V rozšířené verzi metoda využívá všechny typy vazeb zmíněné v kapitole 2.3. Kromě těchto čtyř typů vazeb je možné využít i tzv. procentuální vazbu, která místo kladného potenciálu bere v potaz procentuální dokončení činnosti, z které tato vazba vychází [11]. V rámci rozšíření této metody popíši pouze čtyři základní vazby síťových diagramů AON. Při jejich využití musíme upravit postup výpočtu vpřed a vzad. V ostatních krocích rozšířená metoda postupuje identicky jako v základní variantě.

Výpočet vpřed Určení hodnot počátečních uzlů grafu se nikterak neliší od základní verze MPM, pro které platí $ES_i = 0$ a $EF_i = t_i$. Pro všechny ostatní uzly j nalezneme všechny uzly i, ze kterých vede orientovaná hrana do uzlu j a podle typu vazby vypočteme následující hodnoty:

- **finish to start:** $(EF_i + a_{ij} + t_j)$
- **start to start:** $(ES_i + a_{ij} + t_j)$
- **finish to finish:** $(EF_i + a_{ij})$
- **start to finish:** $(ES_i + a_{ij})$

Poté můžeme určit nejdříve možný konec činnosti EF_j jako maximum z těchto hodnot, dopočíst nejdříve možný začátek činnosti ES_j a ověřit podmínku záporného potenciálu všech vazeb v závislosti na jejím typu:

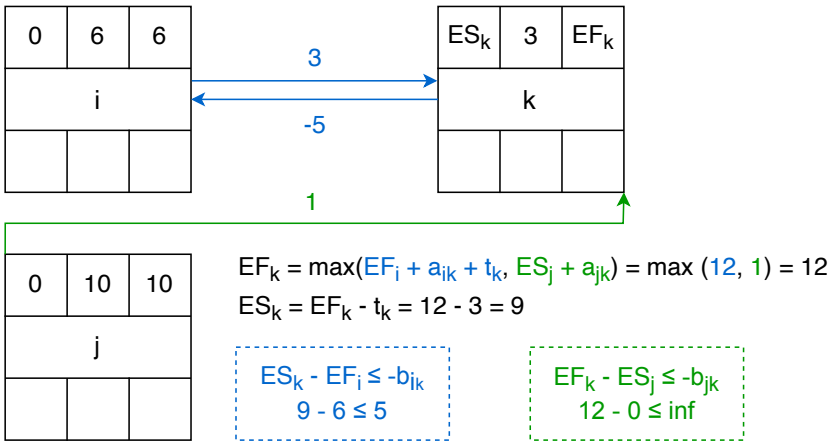
- **finish to start:** $(ES_j - EF_i \leq -b_{ij})$
- **start to start:** $(ES_j - ES_i \leq -b_{ij})$
- **finish to finish:** $(EF_j - EF_i \leq -b_{ij})$
- **start to finish:** $(EF_j - ES_i \leq -b_{ij})$

Výpočet vzad Nejdříve, jako v základní verzi MPM, určíme hodnoty koncových uzlů i $LF_i = max(EF_i)$ a $LS_i = ES_i = LF_i - t_i$. Následně pro všechny ostatní uzly i nalezneme všechny uzly j, do kterých vede orientovaná hrana z uzlu i a podle typu vazby vypočteme následující hodnoty:

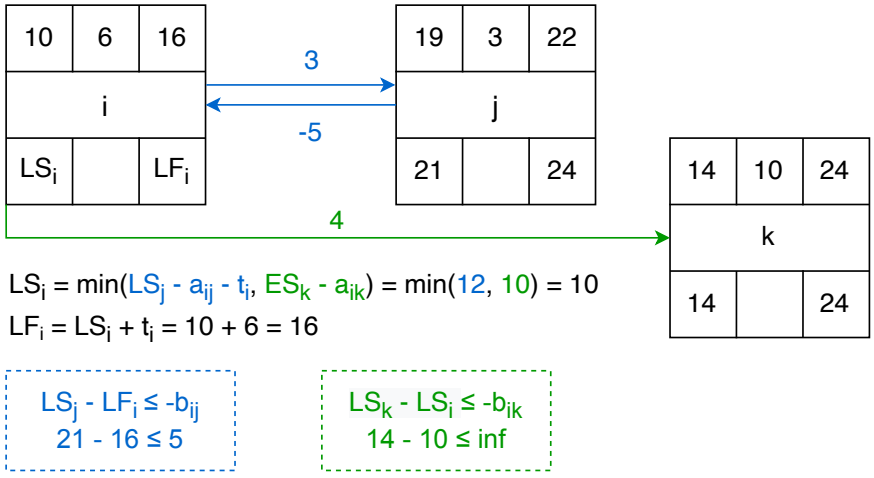
- **finish to start:** $(LS_j - a_{ij} - t_i)$
- **start to start:** $(LS_j - a_{ij})$
- **finish to finish:** $(LF_j - a_{ij} - t_i)$
- **start to finish:** $(LF_j - a_{ij})$

Následně určíme nejpozději možný začátek činnosti LS_i jako minimum z těchto hodnot, dopočteme nejpozději možný konec činnosti LF_i a opět ověříme podmínku záporného potenciálu všech vazeb v závislosti na jejím typu:

- **finish to start:** $(LS_j - LF_i \leq -b_{ij})$
- **start to start:** $(LS_j - LS_i \leq -b_{ij})$
- **finish to finish:** $(LF_j - LF_i \leq -b_{ij})$
- **start to finish:** $(LF_j - LS_i \leq -b_{ij})$



Obrázek 2.8: Příklad výpočtu vpřed rozšířené metody MPM



Obrázek 2.9: Příklad výpočtu vzad rozšířené metody MPM

2.3.4 Metoda PERT

Další metodou pro časovou analýzu projektu je metoda PERT, celým názvem Program Evaluation and Review Technique. Hlavním rozdílem oproti metodám CPM a MPM je ohodnocení síťového diagramu, které je stochastické. Doba trvání činnosti není v této metodě definovaná konstantou, ale náhodnou veličinou s určitým rozdělením pravděpodobnosti. Empiricky bylo zjištěno, že v praxi toto nejlépe vystihuje tzv. beta rozdělení, které lépe vystihuje proměnlivost provozních podmínek (například důlní provoz). Toto rozdělení je velmi blízké rozdělení normálnímu, je spojité, jednovrcholové, mírně asymetrické, ale na rozdíl od normálního je oboustranně ohraničené [12]. Pro pochopení těchto stochastických metod je potřeba znát základy pravděpodobnosti, které nejsou v této práci popsané. Tato metoda tedy nebude dále detailněji popsána.

2.3.5 Metoda GERT

Metoda GERT (Graphical Evaluation and Review Technique) se zabývá časovým plánováním projektů, jejichž průběh se může měnit v závislosti na předešlých výsledcích. Jinými slovy lze projekt uskutečnit vícero způsoby. Stejně jako metoda PERT, tak i metoda GERT je stochastická. Na rozdíl od metody PERT navíc umožňuje použití cyklů a větvení projektu. [13]

Kapitola 3

Analýza a návrh

3.1 Procesy

Cílem této diplomové práce je návrh a následný vývoj výukové webové aplikace, která pomůže studentům pochopit a vyzkoušet si matematické modelování pomocí metody MPM, která se využívá při časovém plánování v projektovém řízení. Důležité je navrhnout intuitivní uživatelské prostředí, se kterým se bude studentům snadno pracovat. Hlavními procesy, které by měla aplikace podporovat jsou:

Seznámení uživatele s metodou Aplikace by měla uživateli poskytnout teoretické informace k MPM metodě. Hlavním podkladem bude teoretická část této diplomové práce, kde je metoda popsána v kapitole 2.3.3. Aplikace bude pracovat s rozšířenou verzí MPM metody o všechny typy vazeb mezi uzly tak, jak je popsáno v kapitole 2.3.3.1

Vysvětlení metody na vzorových příkladech Aplikace poskytne uživateli několik vzorových příkladů, na kterých bude metoda MPM simulována. Uživatel si bude moct zobrazit celý průběh výpočtu této metody. Vzorový příklad bude uživatel moct dále upravovat.

Vizuální editor pro práci s grafy Aplikace bude obsahovat vizuální editor, kde si uživatel bude moct vytvořit vlastní síťový diagram, na který bude aplikována metoda MPM. V tomto editoru bude uživateli umožněno vytvořit, upravit a smazat jednotlivé uzly reprezentující činnosti a vazby mezi nimi.

Zobrazení vzorců na základě vytvořeného grafu Součástí stránky s vizuálním editorem bude přesný popis výpočtu jednotlivých kroků metody MPM včetně všech vzorců použitých při výpočtu.

Nápověda pro tvorbu síťových diagramů Aplikace bude doplněna o tutoriál, ve kterém bude vysvětleno ovládání celého editoru.

Ukládání rozpracovaných příkladů Studentovi bude umožněno si rozpracovaný příklad z aplikace stáhnout ve formátu JSON. Aplikace uživateli umožní tento soubor načíst pro další práci nehledě na to, zda pracuje na stejném zařízení, kde rozpracovaný příklad uložil.

3.2 Funkční požadavky

Funkční požadavky na aplikaci vychází z vydefinovaných procesů v předešlé kapitole. Jedná se o detailnější rozpracování procesů, ze kterých plynou hlavní požadavky na funkcionality aplikace (namapování procesů na funkční požadavky). Pro tuto aplikaci lze požadavky rozdělit do následujících čtyř skupin:

3.2.1 Požadavky na teoretické informace a vzorové příklady

- Aplikace poskytne studentovi teoretické informace o metodě MPM a to především vysvětlení průběhu výpočtu metody, který zahrnuje výpočet vpřed, výpočet vzad, ověření podmínek záporného potenciálu vazeb, výpočet časových rezerv a určení kritické cesty.
- Aplikace poskytne studentovi k nahrání do vizuálního editoru minimálně 3 vzorové příklady, na kterých bude simulována metoda MPM.

3.2.2 Požadavky na vizuální editor a zobrazení výpočtu

- Aplikace umožní vytváření síťových diagramů, na kterých bude simulována metoda MPM.
- Aplikace umožní vytváření a mazání jednotlivých uzlů a hran síťového diagramu.
- Aplikace umožní definovat v rámci uzlů a hran všechny potřebné parametry pro metodu MPM.
- Aplikace umožní jednotlivé uzly síťového diagramu přesouvat (posunutím uzlu se posunou i příslušné hrany).
- Aplikace zobrazí v reálném čase na základě stavu síťového diagramu průběh výpočtu metody MPM a její výsledek, pokud má řešení.
- Aplikace graficky zvýrazní v síťovém diagramu kritickou cestu. Případně všechny kritické cesty, pokud jich je více.

3.2.3 Požadavky na tutoriál

- Aplikace poskytne uživateli všechny potřebné informace ohledně ovládání vizuálního editoru.
- Jednotlivé akce budou popsány za použití textu a názorných obrázků nebo animací.
- Jednotlivé akce budou seskupeny do logických celků (záložek), mezi kterými bude uživatel moct přecházet.
- Na posledním kroku tutoriálu bude moct uživatel přejít do editoru nebo načíst jeden ze vzorových příkladů.

3.2.4 Požadavky na ukládání a nahrávání příkladů

- Aplikace umožní uživateli uložit si rozpracovaný příklad ve vizuálním editoru do souboru ve formátu JSON.
- Aplikace umožní uživateli nahrát rozpracovaný příklad zpět do vizuálního editoru z jakékoliv stránky aplikace.
- Aplikace zobrazí varovnou hlášku v případě, že nahrávaný soubor nebude obsahovat validní data.

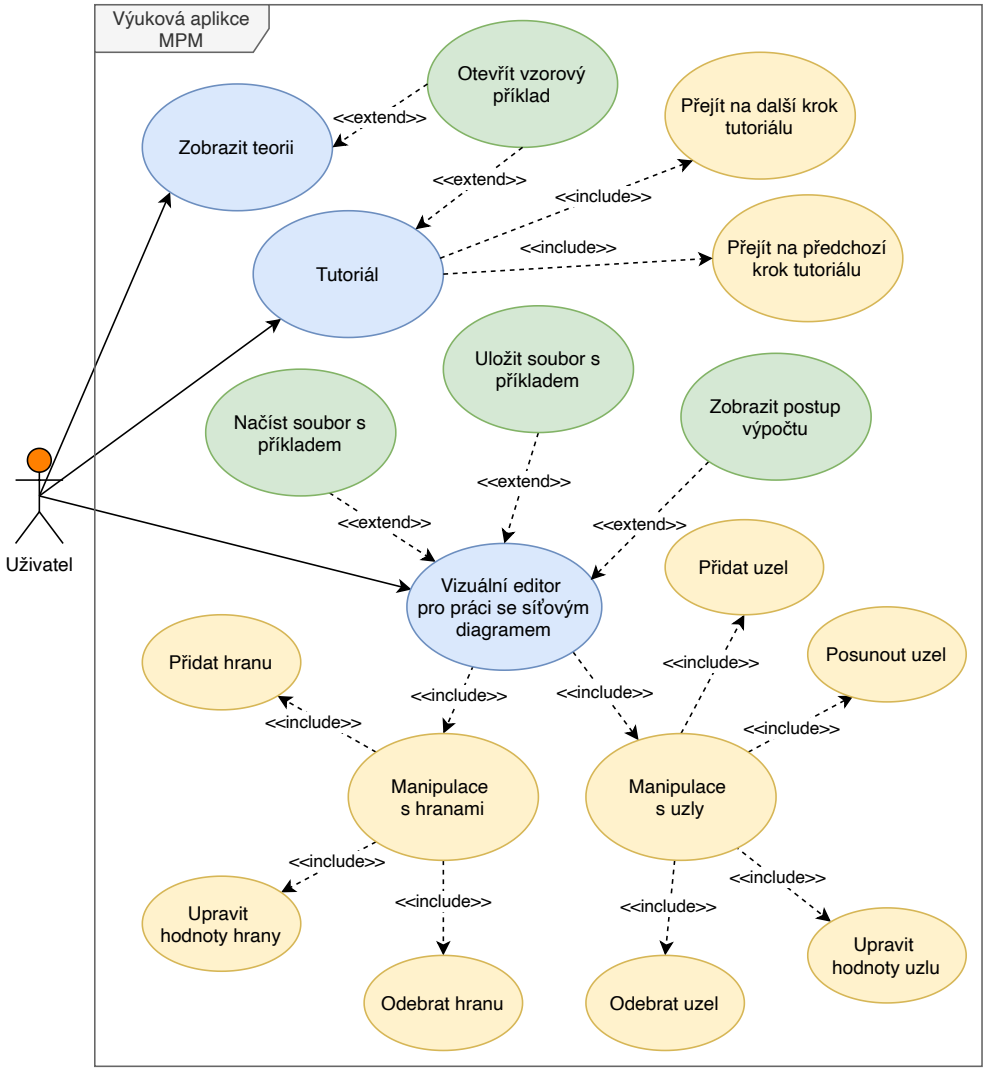
3.3 Nefunkční požadavky

Na základě vydefinování jednotlivých funkčních požadavků a faktu, že se jedná o webovou aplikaci, byly určeny nefunkční požadavky. Tyto požadavky kladou důraz především na dobrou dostupnost napříč různými platformami. Aplikace by měla být implementována tak, aby byla spustitelná jak bez připojení internetu z lokálního úložiště, tak i ze vzdáleného serveru. V budoucnu by mělo být možné ji rozšířit o serverovou část obsahující například další příklady a tutoriály či autentizaci studentů.

- Aplikace bude spustitelná v prohlížeči bez nutnosti instalace.
- Aplikace bude spustitelná lokálně bez nutnosti připojení k internetu.
- Aplikace bude spustitelná z externího serveru, na kterém bude uložena.
- Aplikace bude plně funkční ve všech moderních prohlížečích podporující ECMAScript 2015 (ES6):
  - Chrome verze 58 a vyšší
  - Safari verze 10 a vyšší
  - Firefox verze 54 a vyšší

  - Microsoft Edge verze 14 a vyšší

- Aplikace bude implementována pouze jako client-side
- Aplikace bude implementována v programovacím jazyku JavaScript za použití vhodného frameworku, knihoven a design systému



Obrázek 3.1: UML diagram případů užití

3.4 Případy užití

Z funkčních požadavků na aplikaci lze přímo určit jednotlivé případy užití, neboli procesy interakce mezi uživatelem a aplikací, které vedou k dosažení

požadovaného výsledku. Případy užití (anglicky use cases) se zachycují grafickými diagramy pomocí jazyka UML (obrázek 3.1). Následně jsou hlavní průchody aplikací detailněji rozepsány ve scénářích případů užití. Ty se skládají z krátkého popisu daného případu užití, jejich aktérů, podmínek pro spuštění, hlavních a případných alternativních scénářů a podmínek pro dokončení. Ve všech uvedených scénářích figurují pouze dva aktéři – uživatel a aplikace. Nebudou tedy již uváděni u každého scénáře.

3.4.1 Zobrazení teorie

Případ užití umožňuje uživateli získat základní teoretické znalosti k pochopení Metra potentital metody.

Podmínky pro spuštění: Uživatel se musí nacházet na jedné ze stránek aplikace.

Tabulka 3.1: Hlavní scénář zobrazení teorie

1	Uživatel vybere z hlavní nabídky možnost zobrazit teorii kliknutím na tlačítko s nápisem „Teorie“. **Alternativní:** Pokud se uživatel nenachází na hlavní stránce, tak klikne na tlačítko s nápisem „Teorie“ v horním menu. **Návrat:** 2
2	Aplikace uživateli zobrazí stránku s teoretickými informacemi, které jsou nezbytné k pochopení dané metody.

Podmínky pro dokončení: Uživatel vidí výukové materiály na stránce teorie.

3.4.2 Tutoriál

Případ užití vysvětlí uživateli ovládání vizuálního editoru pro tvorbu síťových diagramů a simulaci MPM metody v několika základních krocích, mezi kterými se může volně pohybovat.

Podmínky pro spuštění: Tento scénář nemá žádné podmínky pro spuštění. Uživatel se může nacházet na jakékoliv stránce aplikace.

Podmínky pro dokončení: Uživatel prošel všemi kroky tutoriálu a nyní se nachází na stránce s vizuálním editorem. Pokud klikl na tlačítko „Otevřít vizuální editor“, tak je vizuální editor prázdný. Pokud klikl na jedno z tlačítek vzorových příkladů, tak je do vizuálního editoru načten zvolený vzorový příklad.

Tabulka 3.2: Hlavní scénář průchodu tutoriálem

1	Uživatel vybere z nabídky na hlavní stránce možnost spustit tutoriál kliknutím na tlačítko s nápisem „Tutoriál“. **Alternativní:** Pokud se uživatel nenachází na hlavní stránce, tak klikne na tlačítko s nápisem „Tutoriál“ v horním menu. **Návrat:** 2 **Alternativní:** Pokud se uživatel nachází na stránce s teorií, může kliknout na tlačítko „Spustit tutoriál“, které se nachází pod textem s teorií. **Návrat:** 2
2	Aplikace uživateli zobrazí náhled ovládacího panelu vizuálního editoru a popis jednotlivých tlačítek.
3	Uživatel klikne na tlačítko „Pokračovat“.
4	Aplikace uživateli zobrazí informace jak ve vizuálním editoru přidávat, mazat a upravovat jednotlivé činnosti (uzly síťového diagramu).
5	Uživatel klikne na tlačítko „Pokračovat“. **Alternativní:** Uživatel klikne na tlačítko „Zpět“. **Návrat:** 2
6	Aplikace uživateli zobrazí informace jak ve vizuálním editoru přidávat, mazat a upravovat jednotlivé vazby (hrany síťového diagramu) mezi činnostmi.
7	Uživatel klikne na tlačítko „Pokračovat“. **Alternativní:** Uživatel klikne na tlačítko „Zpět“. **Návrat:** 4
8	Aplikace uživateli zobrazí informace jak ve vizuálním editoru zobrazit průběh výpočtu MPM metody na aktuálně vytvořeném síťovém diagramu.
9	Uživatel klikne na tlačítko „Otevřít vizuální editor“. **Alternativní:** Uživatel klikne na tlačítko „Zpět“. **Návrat:** 6 **Alternativní:** Uživatel klikne na jedno z tlačítek pro otevření vzorového příkladu.

3.4.3 Manipulace s činnostmi ve vizuálním editoru

Následující scénáře popisují všechny akce týkající se manipulace s činnostmi síťového diagramu.

3.4.3.1 Vytvoření činnosti

Podmínky pro spuštění: Uživatel se musí nacházet na stránce s vizuálním editorem.

Tabulka 3.3: Hlavní scénář vytvoření činnosti

1	Uživatel klikne na tlačítko „Nová činnost“ v ovládacím panelu editoru. **Alternativní:** Uživatel klikne dvakrát myší do prostoru editoru.
2	Aplikace otevře modální okno se vstupními poli pro zadání názvu činnosti a doby jejího trvání. Do vstupního pole pro zadání doby trvání bude možné zadat pouze kladnou číselnou hodnotu nebo nulu.
3	Uživatel vyplní název činnosti a dobu jejího trvání. Následně klikne na tlačítko „Vytvořit“. **Alternativní:** Uživatel klikne na tlačítko „Zrušit“ a tím se scénář ukončí bez změny stavu síťového diagramu.
4	Aplikace ověří, zda byla vyplněna obě textová pole. Modální okno zavře a vytvoří uzel síťového diagramu reprezentující zadanou činnost. Pokud uživatel zahájil scénář kliknutím na tlačítko „Nová činnost“ v ovládacím panelu, tak se uzel vloží doprostřed plochy editoru. V případě dvojkliku se uzel vytvoří v místě kliknutí. **Alternativní:** Aplikace zjistí, že uživatel nevyplnil jedno z požadovaných polí. V takovém případě zobrazí u daného pole varovnou hlášku, že je potřeba příslušnou hodnotu vyplnit. **Návrat:** 3

Podmínky pro dokončení: Ve vizuálním editoru se nachází činnost se zadanými hodnotami.

3.4.3.2 Úprava činnosti

Podmínky pro spuštění: Uživatel se musí nacházet na stránce s vizuálním editorem, ve kterém je vytvořena alespoň jedna činnost (uzel).

Tabulka 3.4: Hlavní scénář upravení činnosti

1	Uživatel klikne dvakrát myší na činnost, kterou chce upravit.
2	Aplikace otevře stejné modální okno jako ve scénáři pro vytvoření činnosti (tabulka 3.6). Aplikace vyplní do pole pro zadání názvu činnosti a doby jejího trvání aktuální hodnoty činnosti.
3	Uživatel upraví stávající hodnoty činnosti na nové. Následně klikne na tlačítko „Uložit“. **Alternativní:** Uživatel klikne na tlačítko „Zrušit“ a tím se scénář ukončí bez změny stavu síťového diagramu.
4	Aplikace ověří, zda byly vyplněny obě textová pole. Modální okno zavře a v činnosti se objeví nově zadané hodnoty. **Alternativní:** Aplikace zjistí, že uživatel nevyplnil jedno z požadovaných polí. V takovém případě zobrazí u daného pole varovnou hlášku, že je potřeba příslušnou hodnotu vyplnit. **Návrat:** 3

Podmínky pro dokončení: Ve vizuálním editoru jsou u upravené činnosti zobrazeny nové hodnoty.

3.4.3.3 Odstranění činnosti

Podmínky pro spuštění: Uživatel se musí nacházet na stránce s vizuálním editorem, ve kterém je vytvořena alespoň jedna činnost (uzel).

Tabulka 3.5: Hlavní scénář odstranění činnosti

1	Uživatel klikne myší na činnost, kterou chce upravit.
2	Aplikace zvýrazní činnost na kterou uživatel klikl.
3	Uživatel zmáčkne klávesu „backspace“.
4	Aplikace danou činnost odstraní ze síťového diagramu včetně všech vazeb souvisejících s danou činností.

Podmínky pro dokončení: Ve vizuálním editoru se nachází síťový diagram bez odstraněné činnosti a souvisejících vazeb.

3.4.4 Manipulace s vazbami činností ve vizuálním editoru

Následující scénáře popisují všechny akce týkající se manipulace s vazbami mezi jednotlivými činnostmi síťového diagramu.

3.4.4.1 Vytvoření vazby

Podmínky pro spuštění: Uživatel se musí nacházet na stránce s vizuálním editorem, ve kterém se nachází síťový diagram minimálně se dvěma činnostmi, mezi kterými není vazba.

Tabulka 3.6: Hlavní scénář vytvoření vazby

1	Uživatel klikne za činnost, ze které má nová vazba (hrana) vystupovat a táhnutím myši přejede na činnost, do které má daná vazba směřovat. V tu chvíli tlačítko myši uvolní.
2	Aplikace ověří, zda již neexistuje vazba mezi těmito činnostmi a zda vytvořením vazby nevznikne v síťovém diagramu smyčka. **Alternativní:** Aplikace detekuje existenci vazby mezi zvolenými činnosti a zobrazí příslušnou varovnou hlášku. Scénář v tomto kroku končí bez změny stavu síťového diagramu. **Alternativní:** Aplikace detekuje, že by při vytvoření vazby mezi zvolenými činnostmi vznikla v síťovém diagramu smyčka a zobrazí příslušnou varovnou hlášku. Scénář v tomto kroku končí bez změny stavu síťového diagramu.
3	Aplikace vytvoří mezi zvolenými činnostmi i a j novou vazbu s výchozími hodnotami, kterými jsou: • Typ vazby: *„finish to start“* (FS) • Kladný potenciál: $a_{i,j} = 0$ • Záporný potenciál: $b_{i,j} = -\infty$

Podmínky pro dokončení: Ve vizuálním editoru se nachází vytvořená vazba mezi činnostmi.

3.4.4.2 Úprava vazby

Podmínky pro spuštění: Uživatel se musí nacházet na stránce s vizuálním editorem, ve kterém se nachází síťový diagram minimálně se dvěma činnostmi a vazbou mezi nimi.

Tabulka 3.7: Hlavní scénář upravení vazby

1	Uživatel klikne dvakrát myší na vazbu, kterou chce upravit.
2	Aplikace otevře modální okno se vstupními poli pro zadání číselných hodnot kladného a záporného potenciálu a pole pro výběr typu vazby. Do vstupních polí pro zadání kladného a záporného potenciálu vazby bude možné zadat pouze číselnou hodnotu. Tato pole bude možné ponechat prázdné, v takové případě bude v poli zobrazena nula pro kladný potenciál a mínus nekonečno pro záporný potenciál.
3	Uživatel upraví hodnoty jednotlivých polí. Následně klikne na tlačítko „Vytvořit“. **Alternativní:** Uživatel klikne na tlačítko „Zrušit“ a tím se scénář ukončí bez změny stavu síťového diagramu.
4	Aplikace ověří, zda zadaný kladný a záporný potenciál splňuje základní podmínku vazby – $a_{i,j} \leq \lvert b_{i,j} \rvert$, kde i je činnost, ze které vede vazba do činnosti j. Modální okno zavře a upraví hodnoty v síťovém diagramu. **Alternativní:** Aplikace zjistí, že uživatel zadal potenciály, které nesplňující jejich základní podmínku uvedenou výše. V takovém případě zobrazí varovnou hlášku, že není tato podmínka splněna. **Návrat:** 3

Podmínky pro dokončení: Ve vizuálním editoru se nachází upravená vazba se zadanými hodnotami.

3.4.4.3 Odstranění vazby

Podmínky pro spuštění: Uživatel se musí nacházet na stránce s vizuálním editorem, ve kterém se nachází síťový diagram minimálně se dvěma činnostmi a vazbou mezi nimi.

Tabulka 3.8: Hlavní scénář odstranění vazby

1	Uživatel klikne myší na vazbu, kterou chce upravit.
2	Aplikace zvýrazní vazbu na kterou uživatel klikl.
3	Uživatel zmáčkne klávesu „backspace“.
4	Aplikace danou vazbu odstraní ze síťového diagramu.

Podmínky pro dokončení: Ve vizuálním editoru se nachází síťový diagram bez odstraněné vazby mezi dvěma činnostmi.

3.4.5 Uložení příkladu do souboru

Případ užití umožňuje uživateli uložit aktuální stav síťového diagramu ve vizuálním editoru pro jeho opětovné načtení.

Podmínky pro spuštění: Uživatel se musí nacházet na stránce s vizuálním editorem, ve kterém je vytvořena alepsoň jedna činnost.

Tabulka 3.9: Hlavní scénář uložení příkladu do souboru

1	Uživatel klikne na tlačítko „Stáhnout“, která se nachází v ovládacím panelu editoru.
2	Aplikace převede aktuální stav síťového diagramu do formátu JSON a zahájí stahování souboru. Při zahájení stahování aplikace zobrazí hlášku „Stahování zahájeno.“.

Podmínky pro dokončení: Soubor s daty o aktuálním stavu vizuálního editoru (síťového diagramu) je stažen do zařízení uživatele.

3.4.6 Načtení příkladu ze souboru

Případ užití umožňuje uživateli načíst dříve uložený soubor z aplikace a obnovit tak předchozí stav vizuálního editoru.

Podmínky pro spuštění: Tento scénář nemá žádné podmínky pro spuštění. Uživatel se může nacházet na jakékoliv stránce aplikace.

Podmínky pro dokončení: Uživatel se nachází na stránce s vizuální editorem ve stavu, ve kterém se nacházel v době, kdy daný soubor uložil.

3.4.7 Simulace metody ve vizuálním editoru

Případ užití umožňuje uživateli vytvářet síťový diagram ve vizuálním editoru. Na základě vytvořených činností a vazeb mezi činnostmi jsou při každé změně dopočítány všechny hodnoty činností a vyznačena v síťovém diagramu kritická cesta jako výsledek metody, pokud výsledku lze dosáhnout. Případ užití dále umožňuje uživateli otevřít okno s průběhem celého výpočtu MPM metody.

Podmínky pro spuštění: Uživatel se musí nacházet na stránce s vizuálním editorem.

Podmínky pro dokončení: Na stránce s vizuálním editorem je zobrazen postup výpočtu MPM metody a jejího řešení. V případě, že metoda nemá řešení, aplikace zobrazí varovnou hlášku s vysvětlením, proč se tak stalo.

Tabulka 3.10: Hlavní scénář načtení příkladu ze souboru

1	Uživatel klikne na tlačítko „Nahrát“, která se nachází v ovládacím panelu editoru. **Alternativní:** Uživatel klikne na tlačítko „Nahrát“, které se nachází v rozcestníku na hlavní stránce. **Alternativní:** Uživatel klikne na tlačítko „Nahrát“, které se nachází v horní liště stránek s teorií, tutoriálem a editorem.
2	Aplikace uživateli zobrazí modální okno s tlačítky: • „Vybrat soubor“ • „Zrušit“ • Zneplatněné tlačítko „Otevřít v editoru“
3	Uživatel klikne na tlačítko „Vybrat soubor“.
4	Aplikace uživateli podle operačního systému otevře nativní okno pro výběr souboru.
5	Uživatel vybere soubor z lokálního úložiště.
6	Aplikace začne zvolený soubor zpracovávat. Ověří, zda se jedná o validní JSON soubor. Mezi tím uživatele informuje zpracovávání souboru.
7	Aplikace zvolený soubor označí jako validní. Vzhledově změní tlačítko „Otevřít v editoru“ jako aktivní. **Alternativní:** Aplikace zvolený soubor označila jako nevalidní. O této skutečnosti informuje uživatele varovnou hláškou. **Návrat:** 3
8	Uživatel klikne na tlačítko „Otevřít v editoru“.
9	Pokud se uživatel nenachází na stránce s vizuálním editorem, tak systém uživatele na tuto stránku přesměruje. Ve vizuálním editoru systém zobrazí síťový diagram z načteného souboru.

Tabulka 3.11: Hlavní scénář simulace metody ve vizuálním editoru

1	Uživatel provede změnu v síťovém diagramu pomocí scénářů věnujících se manipulaci s činnostmi uvedených v kapitole 3.4.3 nebo scénářů věnujících se manipulaci s hranami uvedených v kapitole 3.4.4.
2	Aplikace spustí na pozadí výpočet MPM metody na vytvořeném síťovém diagramu. Po dokončení tohoto výpočtu vyznačí v síťovém diagramu kritickou cestu jako výsledek metody. Pokud některá z vazeb síťového diagramu nesplňuje podmínky záporného potenciálu při výpočtu vpřed nebo vzad, tak je tato vazba označena červeně.
3	Uživatel klikne na tlačítko „Zobrazit výpočet“ z ovládacího panelu editoru.
4	Aplikace zobrazí překryvnou vrstvu, ve které vypíše průběh celého výpočtu: • Výpočet vpřed, při kterém budou pro každou činnost uvedeny následující informace: – informaci zda se jedná o počáteční uzel – výpočet hodnoty Early Start – výpočet hodnoty Early Finish – ověření podmínky záporného potenciálu vazeb • Výpočet vzad, při kterém budou pro každou činnost uvedeny následující informace: – informaci zda se jedná o koncový uzel – výpočet hodnoty Late Start – výpočet hodnoty Late Finish – ověření podmínky záporného potenciálu vazeb • Výpočet časové rezervy pro každou činnost • Jak byla určena kritická cesta U všech výpočtů budou uvedeny vzorce, ze kterých se k výsledné hodnotě došlo. Nesplněné podmínky záporného potenciálu budou označeny červeně.

3.5 Návrh uživatelského rozhraní

Přívětivé uživatelské rozhraní je jedním z nejdůležitějších aspektů dnešních aplikací. Cílem dobrého návrhu uživatelského rozhraní je snadné a intuitivní ovládání aplikace z pohledu uživatele. V ideálním případě takového, aby uživatel zvládl ovládání aplikace bez nutnosti studování uživatelské dokumentace a využívání nápovědy. Při návrhu uživatelského rozhraní se vývojář může řídit například podle 10 bodů Nielsenovy heruistiky [14]:

Visibility of system status Stav aplikace musí být vždy viditelný. Uživatelské rozhraní musí uživatele informovat o stavu, v jakém se aplikace v dané době nachází. Pokud například uživatel čeká na odpověď aplikace, tak by měl být o tomto stavu informován ikonou načítání.

Match between system and the real Word Ovládání aplikace by mělo co nejvíce připomínat práci v reálném světě. Ikonka lupy by měla sloužit pro přiblížení nebo oddálení objektu, nikoliv například pro smazání objektu.

User control and freedom Aplikace musí uživateli dovolit vrátit se do předchozího stavu nebo zrušit zvolenou akci. Pokud se například uživatel rozhodne upravit činnost ve vizuální editoru a následně si to rozmyslí, musí mu být umožněno úpravu zrušit.

Consistency and standards Aplikace by měla být vzhledově konzistentní. Jednotlivé komponenty jako tlačítka, nadpisy nebo vyskakovací okna by měly být stejné napříč celou aplikací. Stejně tak by se neměly měnit popisky identických akcí. Nemělo by se například střídat slovo „uložit“ s „upravit“. Pro zachování konzistence lze využít vhodný design systém, který zaručí konzistenci použitím předdefinovaných komponent.

Error prevention Aplikace by měla přecházet chybovým stavům, například kontrolou zadaných dat před jejich vložením. V případě, že data nejsou validní, měla by uživatele aplikace na tuto skutečnost upozornit a data nezpracovat. Příkladem z této aplikace může být ověření podmínky kladného a záporného potenciálu vazby při její editaci.

Recognition rather than recall Aplikace by měla uživatele co nejméně kognitivně zatěžovat. Uživateli by měla nabízet pouze dostupné akce vzhledem k aktuálnímu stavu aplikace. Ostatní akce by měla aplikace skrýt nebo vizuálně zneplatnit. Naopak informace důležité pro danou část aplikace by měli být vždy přístupné, aby se co nejméně zatěžovala uživatelova paměť.

Flexibility and efficiency of use Ovládání aplikace by mělo být flexibilní a efektivní. Aplikace by měla uživateli umožnit zrychlení procesu ovládáním pomocí klávesových zkratek. Příkladem může být zavření modálního okna klávesou „escape“ nebo smazání objektu klávesou „backspace“ tak, jak je na to většina uživatelů zvyklá z různých aplikací.

Aesthetic and minimalist design Mozek člověka si dokáže zapamatovat pouze omezené množství informací v jeden okamžik. Proto je důležité uživateli zobrazovat jen relevantní prvky a nezatěžovat tak jeho paměť.

Help users recognize, diagnose, and recover from errors Chybové hlášky v aplikaci by měli být v běžném jazyce. Jejich obsah by měl uživatele navést k možnému řešení. Příkladem je validace formulářů a jejich jednotlivých polí nebo v konkrétnějším případě této aplikace informace o tom, v jakém kroku se výpočet zastavil kvůli nesplnění podmínek a o jakou podmínku šlo.

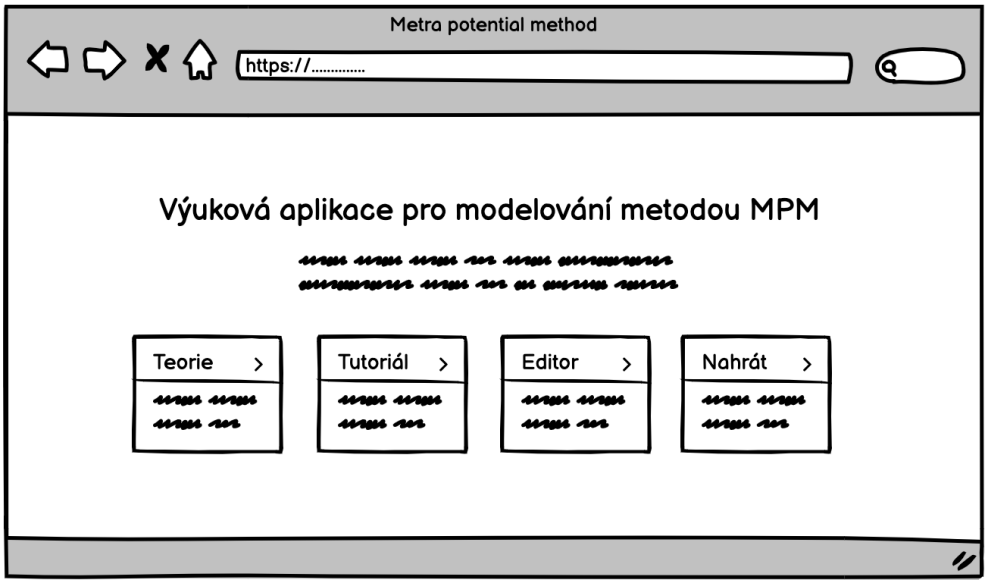
Help and documentation Ve složitějších aplikací se může stát, že uživatel potřebuje nápovědu k provedení nějaké akce. Ta může být buď kontextová nebo globální. V rámci této výukové aplikace bude pro tento bod sloužit část s tutoriálem, ve kterém bude popsáno ovládání vizuálního editoru.

V rámci návrhu uživatelského rozhraní jsem se snažil řídit všemi body této heruistiky. Na základě funkčních požadavků a případů užití jsem aplikaci rozdělil na 4 hlavní obrazovky, které budou popsány níže jak textově, tak graficky za použití wireframů (drátěných modelů), které budou dále sloužit jako podklad při následnou implementaci.

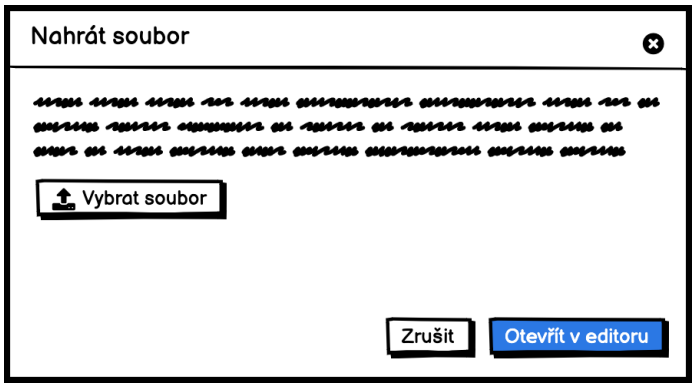
3.5.1 Hlavní stránka

Hlavní stránka webové aplikace bude sloužit jako rozcestník, který umožní uživateli přechod na stránku s teorií, tutoriálem a vizuálním editorem. Dále uživateli umožní nahrát rozpracovaný příklad.

Funkce nahrání bude řešena modálním oknem, které bude stejné napříč celou aplikací – soubor bude možné nahrát na jakékoliv stránce. Toto modální okno bude obsahovat pole pro výběr souboru, tlačítko „Zrušit“ a tlačítko „Otevřít v editoru“, které bude ve výchozím stavu zneplatněné. Po výběru souboru se automaticky soubor začne načítat do paměti prohlížeče. V průběhu nahrávání bude uživateli zobrazována příslušná hláška s ikonkou točícího se kolečka jako indikátor stavu aplikace. Po načtení souboru se graficky označí tlačítko „Otevřít v editoru“ jako aktivní a uživatel pomocí něho bude moci přejít do editoru, který bude ve stavu jako v momentě uložení souboru.



Obrázek 3.2: Wireframe - Hlavní stránka



Obrázek 3.3: Wireframe - Modální okno pro nahrání souboru

3.5.2 Stránka s teorií

Na stránce věnující se teorii MPM metodě budou shrnuty nejpodstatnější informace z teoretické částí této diplomové práce vztahující se k této metodě:

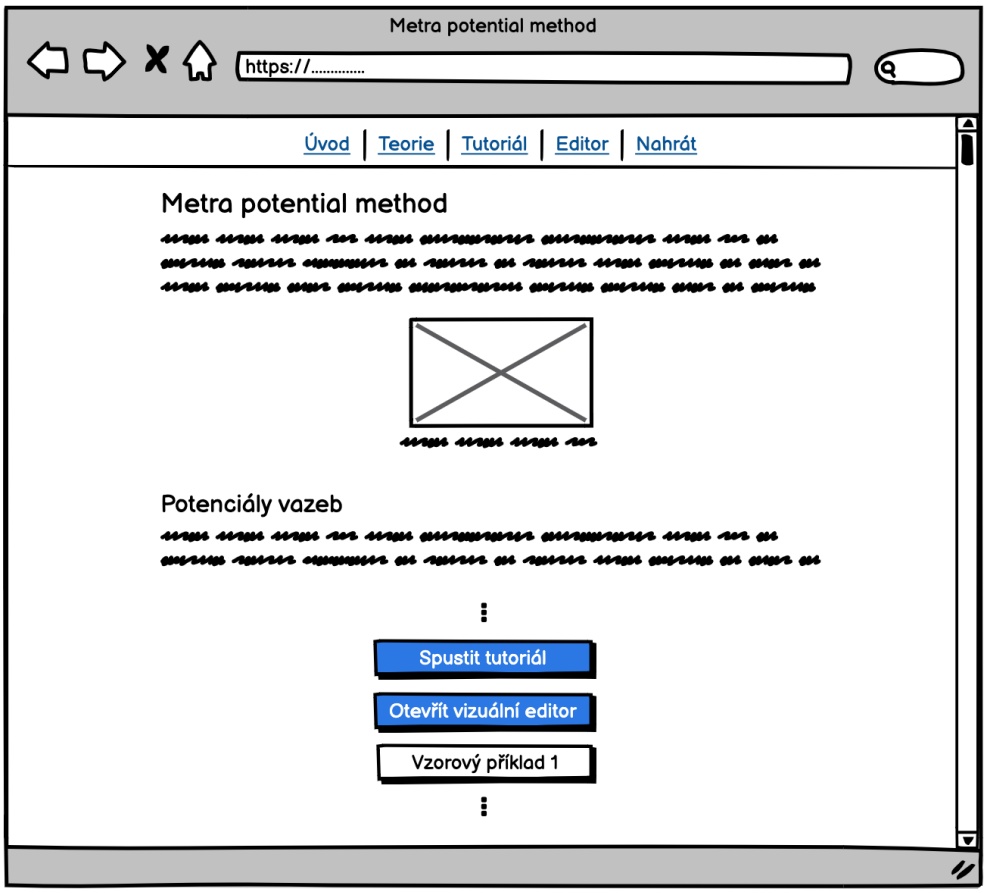
- Úvod k metodě, reprezentace uzlu síťového diagramu (činnosti) a hrany mezi nimi (vazby)
- Potenciály vazeb a jejich podmínky
- Možné typy vazeb
- Průběh výpočtu:

– výpočet vpřed s ověřením podmínky záporného potenciálu

– výpočet vzad s ověřením podmínky záporného potenciálu

– výpočet časových rezerv jednotlivých činností

– určení kritické cesty – výsledku metody



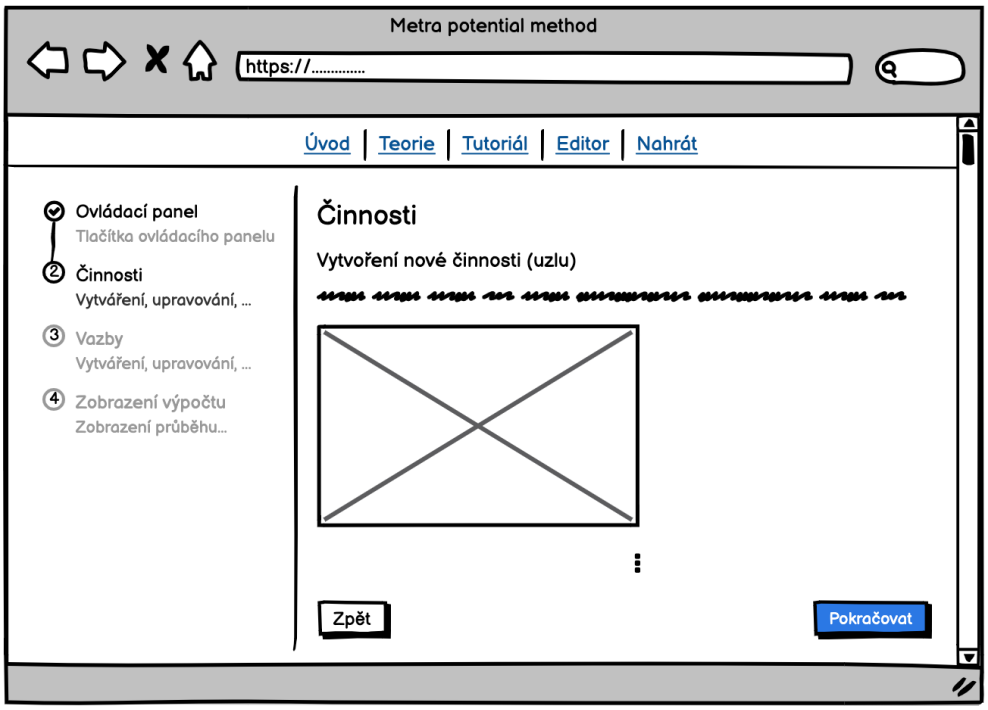
Obrázek 3.4: Wireframe - Stránka s teorií

Bude se tedy jednat o jednoduchou obsahovou stránku využívající základní elementy pro formátování textu. Na rozdíl od hlavní stránky bude tato stránka, i všechny zbylé, obsahovat záhlaví stránky s navigací. Ta uživateli umožní pohybovat se v aplikaci, jako by byl na hlavní stránce s rozcestníkem. Jak již bylo zmíněno, tak tlačítko pro nahrání souboru otevře stejné modální okno, které je použito na hlavní stránce. Pod jednotlivými odstavci s teorií budou tlačítka pro přechod do tutoriálu a vizuální editoru včetně výběru mezi načtením několika vzorových příkladů.

3.5.3 Stránka s tutoriálem

Na této stránce bude uživatel seznámen s ovládáním vizuálního editoru pro tvorbu síťového diagramu a simulace metody MPM. Stránka bude rozdělena celkem na 4 záložky, mezi kterými bude uživatel postupně přecházet:

- Ovládací panel: zde budou popsány všechna tlačítka ovládacího panelu a jejich funkcionalita
- Činnosti: vytváření, upravování a mazání činností (uzlů)
- Vazby: vytváření, upravování a mazání vazeb (hran)
- Zobrazení výpočtu: kde uživatel nalezne kompletní postup výpočtu metody MPM na vytvořeném síťovém diagramu



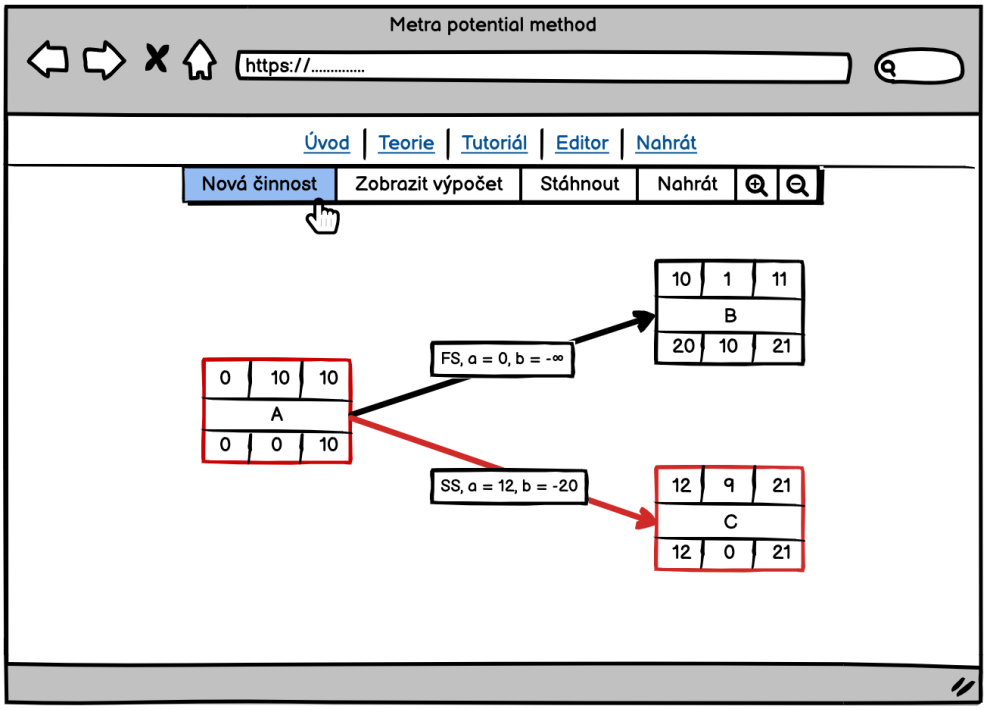
Obrázek 3.5: Wireframe - Stránka s tutoriálem

Popis jednotlivých akcí bude doplněn o pohyblivé obrázky ve formátu gif, na kterých budou dané akce názorně předvedeny. Pod obsahem záložky budou tlačítka pro přechod na předchozí a následující záložku. Kromě toho bude uživatel moci přecházet mezi jednotlivými záložkami pomocí ukazatele kroků tutoriálu v panelu na levé straně od obsahu záložky. Na poslední záložce budou podobně jako na stránce s teorií tlačítka pro přechod do vizuálního editoru včetně výběru mezi načtením několika vzorových příkladů.

3.5.4 Stránka s vizuálním editorem

Na této stránce bude celý prostor okna prohlížeče, kromě horního panelu s navigací a patičky, vyplněn plochou pro editor, ve kterém si uživatel bude moci vytvořit síťový diagram. V horní částí editoru bude ovládací panel s následujícími tlačítky:

- **Nová činnost:** Otevře modální okno s formulářem pro vytvoření nové činnosti, která bude umístěna doprostřed editoru.
- **Zobrazit výpočet:** Otevře okno s detailně popsaným postupem výpočtu.
- **Stáhnout:** Zahájí stahování aktuálního stavu editoru do JSON souboru.
- **Nahrát:** Otevře okno pro nahrání uloženého souboru.
- **Přiblížit:** Přiblíží objekty v editoru.
- **Oddálit:** Oddálí objekty v editoru.



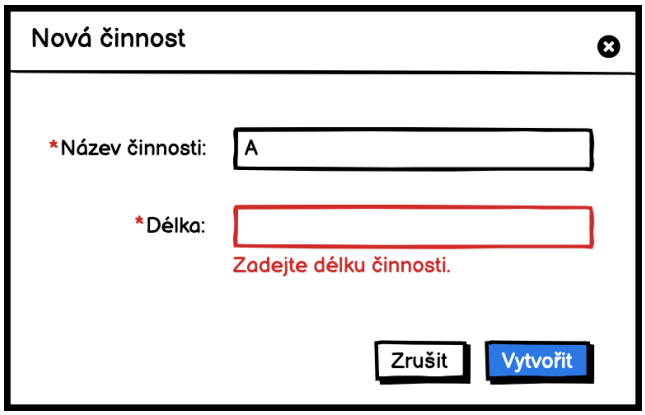
Obrázek 3.6: Wireframe - Stránka s editorem

Prostor editoru bude ve své podstatě neomezený – uživatel se v něm bude moci posouvat táhnutím myši. Jednotlivé činnosti (uzly) budou v editoru vyobrazeny jako na obrázku 2.5 v kapitole 2.3.2. Všechny vazby budou v editoru

znázorněny jako šipky a to pouze typu *„finish to start“*. Hlavním důvodem je zjednodušení implementace a přehlednost pro uživatele, kdy by se v případě grafického znázornění podle obrázku 2.3 začaly činnosti a vazby nepřehledně překrývat. Aplikace i tak bude podporovat všechny typy vazeb mezi činnostmi, pouze bude tento údaj zobrazen v popisu hrany vedle kladného a záporného potenciálu vazby.

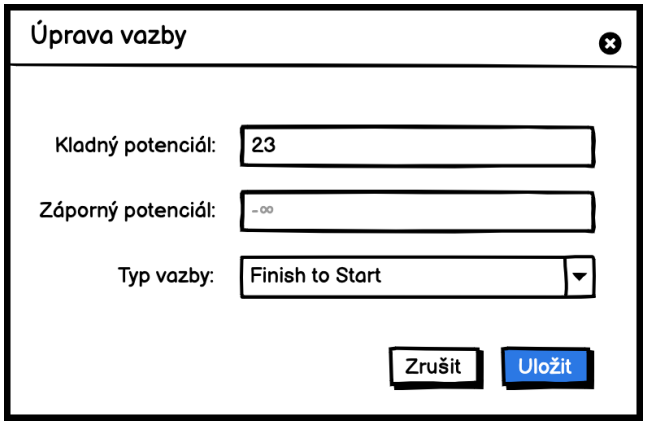
Jednotlivé činnosti bude možné posouvat po ploše editoru tažením myši. Kliknutím na činnost či vazbu se daný objekt označí a bude ho možné smazat klávesou „backspace“. Při dvojím kliknutí na objekt se otevře modální okno pro úpravu jeho hodnot. Dvojím kliknutím do volného prostoru editoru se otevře modální okno pro vytvoření nové činnosti jako při kliknutí na tlačítko „Nová činnost“ v ovládacím panelu. Při jakékoliv změně síťového diagramu bude automaticky simulována metoda MPM. Její výsledek (kritická cesta) bude v diagramu zobrazena červeným zvýrazněním prvků na této cestě.

Modální okno pro vytváření a úpravu činnosti Modální okno pro vytváření a úpravu činnosti (uzlu) bude obsahovat formulář s poli pro zadání názvu činnosti a doby jejího trvání. Pokud se uživatel pokusí uložit činnost bez vyplněné jedné z hodnot, bude upozorněn varovnou hláškou pod příslušným textovým polem. Do pole pro zadání délky doby trvání bude možné zadat pouze kladnou číselnou hodnotu nebo nulu.



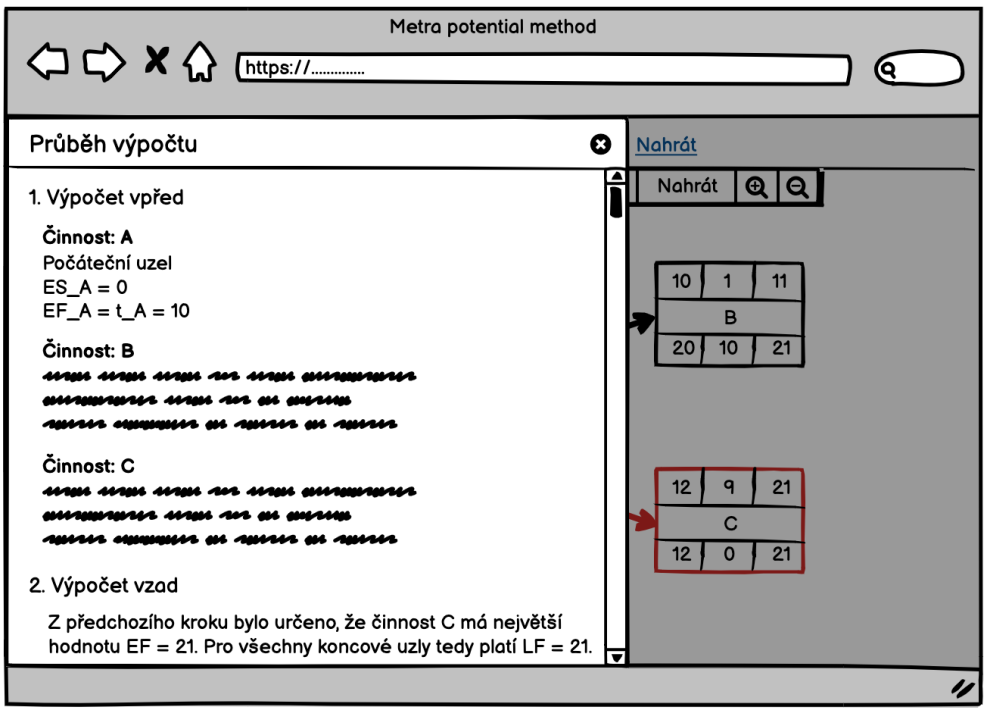
Obrázek 3.7: Wireframe - Modální okno pro vytváření a úpravu činnosti

Modální okno pro úpravu vazby Modální okno pro vytváření a úpravu vazby (hrany) bude obsahovat formulář s poli pro zadání kladného potenciálu, záporného potenciálu a typu vazby. Pole pro typ vazby bude řešeno pomocí pole typu `select`. Při pokusu o uložení změn bude zkontrolována základní podmínka potenciálu vazeb z kapitoly 2.3.3. O jejím nesplnění bude uživatel informován varovnou hláškou a systém uživateli nedovolí takovou vazbu uložit.



Obrázek 3.8: Wireframe - Modální okno pro úpravu vazby

Zobrazení průběhu výpočtu Pro průběh výpočtu bude využita překryvná vrstva, která se zobrazí po kliknutí na tlačítko „Zobrazit výpočet“ na ovládacím panelu editoru. Jeho obsahem bude kompletní postup výpočtu – výpočet vpřed včetně ověření podmínek záporného potenciálu vazeb, výpočet vzad včetně ověření podmínek záporného potenciálu vazeb, výpočet časových rezerv a určení kritické cesty.



Obrázek 3.9: Wireframe - Zobrazení průběhu výpočtu v editoru

3.5.5 Design system

Design system, česky designový systém, je kolekce znovupoužitelných komponent (tlačítka, tabulky, modální okna, apod.), které se řídí jednotným standardem. Tyto kolekce tedy splňují bod „Consistency and standards“ z Nielenovi heuristiky. Z jednotlivých komponent lze sestavit celou aplikaci a urychlit tak její vývoj. V dnešní době tyto systémy využívají největší poskytovatelé aplikací jako Google (Material Design), Shopify (Polaris) nebo Uber (Base Web) a často je poskytují jako open-source.

Při vývoji této aplikace bude použit vhodný design system obsahující všechny potřebné komponenty. Jeho volba bude rozebrána v následující kapitole věnující se implementaci.

Kapitola 4

Implementace

4.1 Zvolené technologie

4.1.1 JavaScript

Celá aplikace je napsána v programovacím jazyce JavaScript. JavaScript [15] je objektově orientovaný programovací jazyk, jehož autorem je Brendan Eich. Svojí syntaxí je podobný programovacím jazykům C, C++ nebo Java, ale zásadně se od nich liší sémanticky. Často je spojován s programovacím jazykem Java, jde však pouze o shodu jmen z marketingových účelů. Ve spojitosti s JavaScriptem se často mluví o pojmu ECMAScript [16], což je jeho standardizovaná verze.

Tento programovací jazyk byl nejprve vyvinut jako doplněk ke značkovacímu jazyku HTML. Byl tedy určen pro webové stránky a jeho interpretaci v tomto případě provádí webový prohlížeč návštěvníka stránky. To znamená, že se program napsaný v JavaScriptu spouští v prohlížeči uživatele po jeho stažení, na rozdíl od jiných interpretovaných programovacích jazyků jako je například PHP nebo Python, které se spouští na straně serveru.

V současné době však existují prostředí jako Node.js, které umožňují běh JavaScriptu i na serverové straně. Implementace na straně serveru klade důraz na vysokou škálovatelnost. Tedy schopnost obsloužit mnoho připojených klientů naráz. Díky této vlastnosti je dnes Node.js velmi oblíbený pro tvorbu API serverů, které dodávají data do webových a mobilních aplikací.

V neposlední řadě se dají pomocí JavaScriptu psát i mobilní aplikace pro operační systémy Android a iOS za využití frameworku React Native [17] či NativeScript [18].

4.1.1.1 TypeScript

Programovací jazyk JavaScript nerozlišuje typy proměnných a návratové hodnoty funkcí programu. To znamená, že v jedné proměnné se může v jednu dobu nacházet číselná hodnota a o chvíli později například pole hodnot. Ty-

peScript [19] je open-source nadstavba JavaScriptu, která ho rozšiřuje o typy a další atributy známé z jiných objektově orientovaných programovacích jazyků:

- anotace typů a typová kontrola
- třídy
- rozhraní
- výčtový typ
- mixiny
- genericita
- moduly
- zkrácená syntaxe pro anonymní funkce
- výchozí hodnoty parametrů funkcí

Typy nám umožňují přidat objektu jasnou definici, zlepšit dokumentaci a udržovat snadněji kód. Díky definici typů může programátor jednodušeji ošetřovat chyby a opravovat je. Pokud se programátor rozhodne využívat TypeScript, je jen na něm, kde všude typování využije. Jinak řečeno je validní JavaScript kód bez úprav i validní TypeScript kód.

4.1.2 React

Stejně jako v jiných programovacích jazycích, tak i v JavaScriptové komunitě vznikají různé frameworky pro usnadnění vývoje aplikací. Framework je softwarová struktura, která slouží jako základ programu a obsahuje řešení pro typické problémy určité oblasti. Mezi nejznámější JavaSriptové frameworky v době psaní této práce patří React, který jsem zvolil pro implementaci výukové aplikace. Mezi hlavní konkurenty Reactu patří například Vue.js nebo Angular.

React [20], známý též jako React.js nebo ReactJS, je open-source framework spravovaný společností Facebook, komunitou vývojářů a firem z celého světa. Může být použit jak pro vývoj webových single-page aplikací, tak pro vývoj mobilních aplikací (React Native). Například webová i mobilní aplikace Facebooku je napsána v tomto frameworku. Dalšími příklady jsou aplikace Instragram, Netflix nebo WhatsApp. Výraz single-page aplikace (SPA) však neznamená, že se jedná pouze o jednostránkovou aplikaci. Na rozdíl od standardních webových stránek dynamicky překresluje jednu stránku novými daty namísto načítání celých nových stránek. Cílem je plynulejší přecházení mezi jednotlivými stránkami aplikace bez efektu přeskakování kvůli načítání všech zdrojů.

Komponenty Hlavním stavebním kamenem aplikací napsaných za využití tohoto frameworku jsou komponenty, ze kterých se postupně skládá celá aplikace. Výhoda těchto komponent je v jejich znovupoužitelnosti. Těmi nejzákladnějšími je například prostý text nebo tlačítko. Z jednotlivých komponent lze poskládat složitější komponenty včetně té hlavní, která tvoří celý obsah aplikace. Komponenty lze vytvářet pomocí funkcí nebo tříd ECMAScript 6:

```
import React from 'react';
import ReactDOM from 'react-dom';

interface HelloProps {
  name: string;
}

const HelloFucntion = (props: HelloProps) => {
  return (<h1>Hello {props.name}</h1>);
}

class HelloClass extends React.Component<HelloProps> {
  return (<h1>Hello {this.props.name}</h1>);
}

function App() {
  return (
    <div>
      <HelloFunction name="Tomas" />
      <HelloClass name="Marie" />
      <HelloFunction name="Petr" />
    </div>
  );
}

ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

Na první pohled se zdá, že návratovou hodnotou komponent je prosté HTML. Jedná se o JSX, neboli JavaScript XML, které je HTML velice podobné značkovacímu jazyku HTML. Poskytuje způsob, jak strukturovat vykreslování komponent pomocí syntaxe známé všem vývojářům webových aplikací. Využití JSX ale není podmínkou. Komponenty mohou být také psány v čistém JavaSciptu.

4.2 Použité knihovny

4.2.1 React router

V kapitole 4.1.2 věnující se React frameworku jsem zmínil, že se pomocí tohoto frameworku vytváří single-page aplikace. Knihovna React router [21] obsahuje komponenty pro implementaci vícestránkového obsahu se zachováním výhod SPA. To znamená, že na základě zadané url adresy uživatelovi vykresluje specifický obsah, ačkoliv se uživatel nachází stále na jedné stránce a nedochází k jejímu opětovnému načtení. V rámci této práce je knihovna použita pro přecházení mezi domovskou stránkou, teorií, tutoriálem a vizuálním editorem:

```
<HashRouter>
    <Switch>
        <Route exact path="/" component={Home} />
        <Route exact path="/teorie" component={Theory} />
        <Route exact path="/tutorial" component={Tutorial} />
        <Route exact path="/editor" component={Editor} />
    </Switch>
</HashRouter>
```

V uvedeném příkladu je využita komponenta `HashRouter`. Ta určuje, že url adresy budou obsahovat znak `#` – z webových stránek známý pro definování takzvané kotvy. Díky tomuto způsobu směrování prohlížeč bere všechny stránky jako jednu a aplikace je funkční i bez serveru (lze ji jednoduše spustit otevřením souboru `index.html`). Alternativou je komponenta `BrowserRouter`, která využívá History API [22] a url adresy se díky ní tváří jako odlišné stránky. K tomu je zapotřebí, aby aplikace běžela na webovém serveru, který bude všechny požadavky směřovat na hlavní soubor `index.html`.

Na jednotlivé stránky se lze v aplikaci odkazovat pomocí komponenty `Link` nebo zavoláním metody `push` hooku `useHistory()`:

```
import { Button } from 'antd';
import { Link, useHistory } from 'react-router-dom';

const Navigation = () => {
  const history = useHistory();

  return (
    <>
        <Link to="/">Úvod</Link>
        <Button onClick={() => history.push('/editor')}>Editor</Button>
    </>
  );
};
```

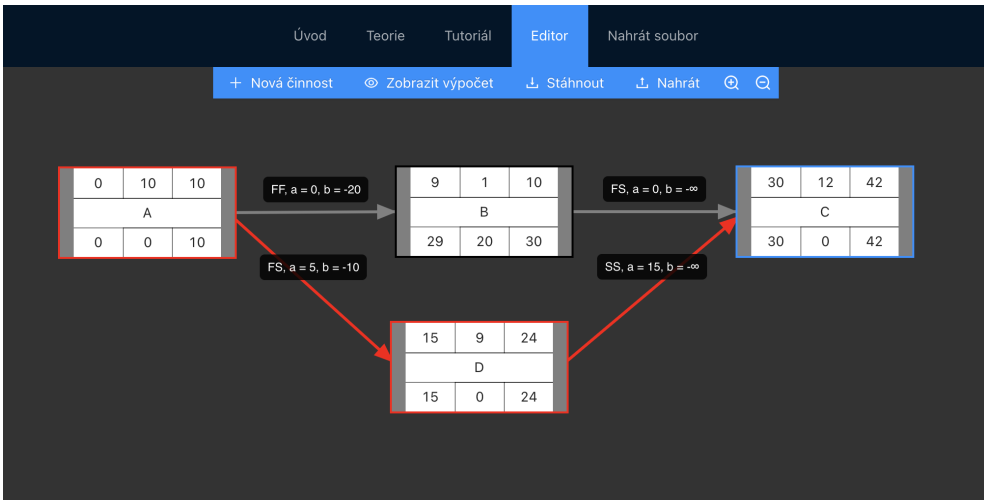
4.2.2 Ant Design

V kapitole 3.5.5 jsem se zmínil o výhodách design systémů. Pro vývoj výukové aplikace jsem zvolil Ant Design [23] na základě velké komunity vývojářů a především kvůli tomu, že obsahuje všechny potřebné prvky využívané v aplikaci. Knihovna je vyvíjena přímo pro React framework, takže jsou všechny prvky dostupné jako React komponenty. Knihovna poskytuje komponenty pro rozvržení stránky, typografii, jednoduché prvky jako tlačítka a ikony až po ty komplexnější, jako jsou modální okna či formuláře.

4.2.3 React diagrams

Stěžejním bodem implementace bylo vyřešit vykreslování a manipulaci se síťovými diagramy. Jednou z možností bylo vykreslování diagramů pomocí SVG značkovacího jazyka a naprogramování vlastní manipulace s jednotlivými prvky. Implementace celého řešení by však zabrala mnoho času. Proto jsem se rozhodl pro využití open-source knihovny React diagrams [24], která splňuje všechny požadavky z návrhu aplikace.

Knihovna umožňuje vytváření vlastních komponent děděním ze základních tříd, které jsou popsané v kapitole 4.3.1. Díky této vlastnosti ji lze přizpůsobit k požadavkům této aplikace. Kromě klasických operací s uzly a hranami grafu umožňuje uživateli například pohybovat se po nekonečném prostoru pracovní plochy a přibližovat či oddalovat síťový diagram. V rámci implementace usnadňuje i načítání a ukládání síťového diagramu do souboru díky metodám pro serializaci a deserializaci, které se dají přizpůsobit v rámci implementace vlastních tříd pro jednotlivé entity. Knihovna disponuje vlastním řešením událostí, díky kterým lze vyvolávat výpočet MPM metody na základě změny jednotlivých uzlů a hran.



Obrázek 4.1: Stránka s vizuálním editorem

4.3 Příklady z implementace

4.3.1 Vizuální editor

Jak již bylo uvedeno v předchozí kapitole 4.2.3, hlavním stavebním kamenem vizuálního editoru je knihovna `react-diagrams`. V rámci této knihovny bylo potřeba vytvořit nové třídy, které jsou odvozeny (dědí) ze základních tříd této knihovny. V následících příkladech popíši jednotlivé třídy vztahující se k uzlům síťového diagramu reprezentující činnosti. Identické třídy bylo potřeba vytvořit i pro hrany, jejich štítky a porty – vstupní a výstupní místa uzlů, na které jsou hrany připojeny.

TaskNodeFactory Továrničky se starají o vytvoření modelu entity a jejího vykreslení. Jejich obsahem jsou dvě metody:

- **generateReactWidget**: metoda vracející React komponentu – tedy objekt, který je zobrazen uživateli
- **generateModel**: metoda vracející nový model příslušné entity

```
/**
 * Task node factory
 */
export class TaskNodeFactory extends
    AbstractReactFactory<TaskNodeModel, DiagramEngine> {
   constructor() {
      super('task');
   }

   /**
    * Generates new link model
    * @param event: GenerateModelEvent
    */
   generateModel(event: GenerateModelEvent): TaskNodeModel {
      return new TaskNodeModel();
   }

   /**
    * Generates JSX component for node
    * @param event
    */
   generateReactWidget(
       event: GenerateWidgetEvent<TaskNodeModel>
    ): JSX.Element {
      return <TaskNodeWidget engine={this.engine} node={event.model} />;
   }
}
```

TaskNodeModel V modelech jsou uchovávána veškerá data o příslušné entitě. V rámci uzlu jsou definovány následující proměnné:

- **name: string** - název činnosti
- **duration: number** - doba trvání
- **earlyStart: number** - hodnota Early Start
- **earlyFinish: number** - hodnota Early Finish
- **lateStart: number** - hodnota Late Start
- **lateFinish: number** - hodnota Late Finish
- **isCritial: boolean** - příznak, zda je uzel zahrnut do kritické cesty
- **reserve: number** - časová rezerva

Všechny proměnné mají úroveň přístupnosti `protected`, takže k nim jsou vytvořeny příslušné gettery a settery. Jednotlivé modely obsahují metody `serialize()` a `deserialize(event: DeserializeEvent<this>)` starající se o mapovaní modelu do objektu pro uložení a načtení ze souboru. Dále jsem model rozšířil o následující metody, které se využívají v rámci simulace MPM metody:

- **isStartNode(): boolean** - vrací příznak, zda se jedná o počáteční uzel
- **isEndNode(): boolean** - vrací příznak, zda se jedná o koncový uzel
- **getOutTaskLinks(): TaskLinkModel**[] - vrací všechny vazby, které směřují z uzlu a mají nastavený cílový uzel
- **getInTaskLinks(): TaskLinkModel**[] - vrací všechny vazby, které směřují do uzlu

TaskNodeWidget Tato třída je klasikou komponentou React frameworku. Stará se o vykreslení objektu uživateli na základě daného modelu, který jí je předán jako parametr.

Celkem bylo naimplementováno 12 tříd pro vytvoření požadovaných entit:

- Uzly (činnosti):
  - TaskNodeFactory
  - TaskNodeModel
  - TaskNodeWidget
- Hrany (vazby):

  - TaskLinkFactory
  - TaskLinkModel
  - TaskLinkWidget
- Štítky hran (popisy vazeb):
  - TaskLabelFactory
  - TaskLabelModel
  - TaskLabelWidget
- Port (vstupní a výstupní body uzlů):
  - TaskPortFactory
  - TaskPortModel

Instance diagramu Základní třídou diagramu je `DiagramEngine`, která se stará o vytváření jednotlivých entit a vykreslení. Do této třídy jsou zaregistrovány všechny továrničky pomocí příslušných metod:

```
const engine = createEngine();
engine.getPortFactories().registerFactory(new TaskPortFactory());
engine.getNodeFactories().registerFactory(new TaskNodeFactory());
engine.getLinkFactories().registerFactory(new AdvancedLinkFactory());
engine.getLabelFactories().registerFactory(new AdvancedLabelFactory());
```

Jednotlivé entity jsou uchovávány ve třídě `DiagramModel`, jejíž instance je nastavena do třídy `DiagramEngine` pomocí metody `setModel(model: CanvasModel)`:

```
const model = new DiagramModel();
engine.setModel(model);

// Task `A` with duration 10
const taskA = new TaskNodeModel();
taskA.setName('A');
taskA.setDuration(10);

// Task `B` with duration 10
const taskB = new TaskNodeModel();
taskB.setName('B');
taskB.setDuration(20);

// Link between task `A` and `B`
let linkAB = new AdvancedLinkModel();
linkAB.setSourcePort(taskA.getPort('right'));
linkAB.setTargetPort(taskB.getPort('left'));

// Adds all entities to diagram model
model.addAll(taskA, taskB, linkAB);
```

4.3.2 Výpočet metody MPM

Výpočet metody se v rámci vizuálního editoru spouští při každé změně stavu síťového diagramu. Tedy pokud dojde k přidání, úpravě či smazání činnosti nebo vazby. Celý výpočet metody začíná voláním funkce `const runMPM = (model: DiagramModel): void`. Tato funkce postupně prochází všechny kroky výpočtu.

```
/**
 * Simulates MPM method on given DiagramModel.
 * @param model: DiagramModel
 */
export const runMPM = (model: DiagramModel): void => {
  let allNodes = model.getNodes().map(node => node as TaskNodeModel);
  let allLinks = model.getLinks().filter(link => !!link.getTargetPort())
    .map(link => link as TaskLinkModel);
  let startNodes = [];
  let endNodes = [];
  for (let node of allNodes) {
    if (node.isStartNode() === true) {
      startNodes.push(node);
    }
    if (node.isEndNode() === true) {
      endNodes.push(node);
    }
    node.resetValues();
  }
  resetLinks(allLinks);
  initForwardCalculation(startNodes);
  initBackwardCalculation(endNodes);
  setReserves(allNodes);
  if (checkConditions(allLinks) === true) {
    findCriticalPaths(startNodes);
  }
}
```

Určení počátečních a koncových uzlů Nejdříve jsou nalezeny všechny počáteční a koncové uzly. Tedy uzly, které mají vstupní nebo výstupní stupeň roven nule. Ke všem uzlům síťového diagramu lze přistoupit pomocí metody `getNodes()` objektu `DiagramModel`, který je vstupním parametrem výpočetní funkce. Následně algoritmus prochází pomocí cyklu všechny uzly a kontroluje počet vstupních a výstupních hran. Na konci tohoto kroku jsou vytvořeny dvě pole – jedno s počátečními uzly a druhé s koncovými uzly. Tyto pole jsou využity při výpočtu vpřed a vzad. V rámci tohoto kroku jsou nastaveny hodnoty uzlů a hran na výchozí hodnoty.

Výpočet vpřed Výpočet vpřed je realizován pomocí algoritmu prohledávání do hloubky využívající rekurzi. Nejdříve je pomocí cyklu u všech počátečních uzlů nastavena hodnota Early Start na nulu a hodnota Early Finish na dobu trvání činnosti. Následně je na tyto uzly zavolána rekurzivní funkce `const forwardCalculation = (node: TaskNodeModel): void`. Tato funkce nalezne všechny sousedící uzly, do kterých vede orientovaná hrana z uzlu `node`, který je vstupním parametrem této funkce. Pro všechny tyto uzly je vypočítána potencionální hodnota Early Finish na základě typu vazby a kladného potenciálu vazby. Pokud je tato hodnota větší než dosavadní hodnota tohoto uzlu, tak je v uzlu uložena, je dopočtena hodnota Early Finish a výpočet pokračuje dalším voláním funkce `computeForward`. Pokud tato hodnota není vyšší, tak je tato část rekurze ukončena.

```
/**
 * Inits forward calculation.
 * @param startNodes: TaskNodeModel[]
 */
const initForwardCalculation = (startNodes: TaskNodeModel[]): void => {
  for (let node of startNodes) {
    node.setEarlyStart(0);
    node.setEarlyFinish(node.getDuration());
    forwardCalculation(node);
  }
}

/**
 * Recursive function for forward calculation.
 * @param node: TaskNodeModel
 */
const forwardCalculation = (node: TaskNodeModel): void => {
  for (let link of node.getOutTaskLinks()) {
    let targetNode = link.getTargetTaskNode();
    let earlyFinish = link.getPossibleEarlyFinish();
    if (earlyFinish > targetNode.getEarlyFinish()) {
      targetNode.setEarlyFinish(earlyFinish);
      targetNode.setEarlyStart(earlyFinish - targetNode.getDuration());
      forwardCalculation (targetNode);
    }
  }
}
```

Výpočet vzad Výpočet vzad je velice podobný výpočtu vpřed. Nejdříve je zjištěna maximální hodnota Early Finish, která je následně nastavena u všech koncových uzlů, na které je zavolána rekurzivní funkce `const backwardCalculation = (node: TaskNodeModel): void`. Tato funkce se snaží nalézt nejmenší hodnotu Late Start a na základě ní vypočíst příslušnou hodnotu Late Finish.

```
/**
 * Finds maximal early finish from given nodes.
 * @param endNodes
 */
const findMaxEarlyFinish = (endNodes: TaskNodeModel[]): number => {
  return Math.max(...endNodes.map(node => node.getEarlyFinish()));
}

/**
 * Inits backward calculation.
 * @param endNodes: TaskNodeModel[]
 */
const initBackwardCalculation = (endNodes: TaskNodeModel[]): void => {
  const maxEarlyFinish = findMaxEarlyFinish(endNodes);
  for (let node of endNodes) {
    node.setLateFinish(maxEarlyFinish);
    node.setLateStart(maxEarlyFinish - node.getDuration());
    backwardCalculation(node);
  }
}

/**
 * Recursive function for backward calculation.
 * @param node: TaskNodeModel
 */
const backwardCalculation = (node: TaskNodeModel): void => {
  for (let link of node.getInTaskLinks()) {
    let sourceNode = link.getSourceTaskNode();
    const lateStart = link.getPossibleLateStart();
    if (sourceNode.getLateStart() > lateStart) {
      sourceNode.setLateStart(lateStart);
      sourceNode.setLateFinish(lateStart + sourceNode.getDuration());
      backwardCalculation(sourceNode);
    }
  }
}
```

Výpočet časových rezerv V tomto kroku je pomocí cyklu pro každý uzel diagramu vypočítána časová rezerva jako rozdíl hodnot Late Start a Early Start. Tato hodnota je uložena jako parametr uzlu.

```
/**
 * Sets time reserve to all nodes based on their values.
 * @param nodes
 */
const setReserves = (nodes: TaskNodeModel[]): void => {
  for (let node of nodes) {
    node.setReserve(node.getLateStart() - node.getEarlyStart());
```

```
  }
}
```

Ověření podmínek záporného potenciálu Ověření podmínek záporného potenciálu jednotlivých vazeb není řešeno v rámci výpočtu vpřed a vzad z toho důvodu, že v daný moment algoritmus neví, zda byla pro daný uzel nalezena výsledná hodnota Early Finish či Late Start. Ověření podmínek probíhá pomocí iterace nad všemi vazbami síťového diagramu, ke kterým lze přistoupit pomocí metody `getLinks()` objektu `DiagramModel`. Pro každou vazbu je ověřena příslušná podmínka pro výpočet vpřed i vzad na základě typu vazby a záporného potenciálu dané vazby. Pokud podmínka není splněna, tak je vazba označena jako chybná nastavením parametru `error` na hodnotu `true`. Vizuální editor na základě této hodnoty následně zvýrazní vazbu červenou barvou.

```
/**
 * Checks negative potential conditions.
 * @param links: TaskLinkModel[]
 */
const checkConditions = (links: TaskLinkModel[]): boolean => {
  let hasError = false;
  for (let link of links) {
    if (link.checkNegativePotentialConditions() === false) {
      link.setError(true);
      hasError = true;
    }
  }
  return !hasError;
}
```

Nalezení kritické cesty Pokud jsou splněny podmínky jednotlivých vazeb, které byly ověřeny v předešlém kroku, tak je zahájeno hledání kritických cest. To je opět řešeno pomocí rekurze. Nejdříve byla kritická cesta hledána pouze pomocí časových rezerv jednotlivých funkcí. Na základě uživatelského testování, které je uvedeno v následující kapitole 5 bylo zjištěno, že je potřeba zahrnout i podmínku týkající se hodnoty Early Finish. Aby byla zahrnuta vazba do kritické cesty, tak musí být její potenciální hodnota Early Finish cílového uzlu shodná s jeho hodnotou po výpočtu vpřed.

```
/**
 * Inits finding of all critical paths.
 * @param startNodes: TaskNodeModel[]
 */
const findCriticalPaths = (startNodes: TaskNodeModel[]) => {
  for (let node of startNodes) {
    if (node.getReserve() === 0) {
```

```
      setCritical(node);
    }
  }
}

/**
 * Sets node as critical and continues in finding critical path.
 * @param node: TaskNodeModel
 */
const setCritical = (node: TaskNodeModel) => {
  node.setIsCritical(true)
  for (let link of node.getOutTaskLinks()) {
    let targetNode = link.getTargetTaskNode();
    let earlyFinish = link.getPossibleEarlyFinish();
    if (targetNode.getReserve() === 0
        && targetNode.getEarlyFinish() === earlyFinish) {
      link.setIsCritical(true);
      setCritical(targetNode);
    }
  }
}
```

Kapitola 5

Testování

Aplikace bude vzdáleně otestována se třemi nezávislými osobami. Cílem uživatelského testování je odhalení skrytých chyb, které mohli být přehlédnuty při vývoji aplikace a případné zlepšení uživatelského rozhraní, které pro reálné uživatele může být nepřehledné nebo složité. Pro účely testování bude aplikace nasazena na webový server, aby si ji uživatel nemusel stahovat do svého zařízení. Samotné testování bude rozděleno na 3 fáze – vstupní dotazník, testovací scénář a výstupní dotazník. Testování se všemi uživateli bude provedeno online za použití komunikačních aplikací s možností sdílení obrazovky.

5.1 Vstupní dotazník

Cílem vstupního dorazníku (pretestu) je získat informace o uživateli, se kterým bude testování prováděno. Uživatel budou položeny následující otázky:

- Jaké používáte zařízení (procesor, operační systém)?
- V jakém prohlížeči budete aplikaci spouštět?
- Studujete nebo vystudoval jste školu s ekonomickým zaměřením?
- Máte nějakou zkušenost s projektovým řízením a metodami pro časové plánování projektu?
- Znáte metodu kritické cesty?
- Znáte metodu Metra pothential method?

5.2 Testovací scénář

1. Otevřete si aplikaci na adrese `http://stage.vaclavdvorak.cz/mpm/`

2. Otevřete stránku s teorií a nastudujte si základní informace o dané problematice. Pokud máte zkušenosti s metodou MPM, přejděte na krok číslo 2.
3. Přejděte na stránku s tutoriálem a prostudujte jednotlivé funkce vizuálního editoru.
4. Otevřete jeden ze vzorových příkladů.
5. Přidejte novou činnost s názvem Test a dobou trvání 20.
6. Vytvořte vazbu mezi libovolnou činností a činností Test.
7. Uložte si rozpracovaný příklad do souboru.
8. Upravte hodnoty vámi vytvořené vazby na kladný potenciál 5 a záporný potenciál -20.
9. Smažte libovolnou činnost.
10. Smažte libovolnou vazbu.
11. Nahrajte vámi uložený soubor z bodu 7, ve kterém jste si soubor uložili.
12. Prostudujte průběh výpočtu.
13. Pokud není splněna některá z podmínek, upravte síťový diagram.
14. Určete výsledek metody.

5.3 Výstupní dotazník

Cílem výstupního dotazníku (posttestu) je zjistit uživatelovy dojmy po průchodu stanoveného scénáře. Uživateli budou položeny následující otázky:

- Narazili jste v průběhu testování na nějaké problémy?
- Co se vám na aplikaci nelibí, co byste zlepšili?
- Co vás na aplikaci zaujalo?
- Pochopili jste problematiku, kterou se výuková aplikace snaží vysvětlit?

5.4 Testování s uživateli

5.4.1 První uživatel

Testování s uživatelem bylo provedeno za použití aplikace Google Meet se zapnutým sdílením obrazovky uživatele.

Vstupní dotazník

- **Jaké používáte zařízení (procesor, operační systém)?**
  Notebook HP EliteBook 840 G3, Intel core i7-6500U, Windows 10
- **V jakém prohlížeči budete aplikaci spouštět?**
  Chrome verze 89.0.4389.128
- **Studujete nebo vystudoval jste školu s ekonomickým zaměřením?**
  Nestudoval ani nestuduje.
- **Máte nějakou zkušenost s projektovým řízením a metodami pro časové plánování projektu?**
  Má zkušenosti z práce jako projektový manažer.
- **Znáte metodu kritické cesty?**
  Ano.
- **Znáte metodu Metra pothential method?**
  Ne.

Průběh testování Uživatel na krátkou dobu zaváhal u 6. bodu testovacího scénáře, kdy se snažil napojit vazbu na činnost. Místo propojení vazby mezi porty činností táhl myší do středu činnosti, což nevedlo k propojení vazby s činností. Tuto situaci vzápětí vyřešil po tom, co si činnost posunul a zjistil, že vazba není správně připojena.

Výstupní dotazník

- **Narazili jste v průběhu testování na nějaké problémy?**
  Pokud vytvořím novou činnost pomocí tlačítka z ovládacího panelu a následně vytvořím další novou činnost stejným způsobem, tak se činnosti překrývají. Dále lze vytvořit vazbu do neurčita.
- **Co se vám na aplikaci nelibí, co byste zlepšili?**
  Aplikace se uživateli líbila a neměl žádné připomínky.
- **Co vás na aplikaci zaujalo?**
  Jednoduché tvoření síťových diagramů. Přehlednost celé aplikace.
- **Pochopili jste problematiku, kterou se výuková aplikace snaží vysvětlit?**
  Uživatel věří, že jí v rámci krátkého testování pochopil.

5.4.2 Druhý uživatel

Testování s uživatelem bylo provedeno za použití aplikace Discord se zapnutým sdílením obrazovky uživatele.

Vstupní dotazník

- **Jaké používáte zařízení (procesor, operační systém)?**
  Desktop PC, AMD Ryzen 5 3600 6-core, Windows 10
- **V jakém prohlížeči budete aplikaci spouštět?**
  Firefox verze 88.0
- **Studujete nebo vystudoval jste školu s ekonomickým zaměřením?**
  Nestudoval ani nestuduje.
- **Máte nějakou zkušenost s projektovým řízením a metodami pro časové plánování projektu?**
  Nemá žádné zkušenosti s projektovým řízením.
- **Znáte metodu kritické cesty?**
  Ne.
- **Znáte metodu Metra pothential method?**
  Ne.

Průběh testování Uživatel splnil všechny úkoly testovacího scénáře bez zaváhání.

Výstupní dotazník

- **Narazili jste v průběhu testování na nějaké problémy?**
  Na žádné problémy při plnění jednotlivých úkolů uživatel nenarazil.
- **Co se vám na aplikaci nelibí, co byste zlepšili?**
  Uživatel by ocenil možnost změnit si barvu vizuálního editoru a jednotlivých činností. Dále by se mu líbilo, pokud by u jednotlivých hodnot činností byl po najetí myši zobrazen popisek s jejich významem.
- **Co vás na aplikaci zaujalo?**
  Jednoduchost ovládání celé aplikace.
- **Pochopili jste problematiku, kterou se výuková aplikace snaží vysvětlit?**
  Uživatel problematiku okrajově pochopil, nicméně uvedl, že by teorii potřeboval důkladněji pročíst.

5.4.3 Třetí uživatel

Testování s uživatelem bylo provedeno za použití aplikace Google Meet se zapnutým sdílením obrazovky uživatele.

Vstupní dotazník

- **Jaké používáte zařízení (procesor, operační systém)?**
  Apple Macbook Air, M1, macOS Big Sur
- **V jakém prohlížeči budete aplikaci spouštět?**
  Chrome verze 89.0.4389.128
- **Studujete nebo vystudoval jste školu s ekonomickým zaměřením?**
  Dokončené magisterské studium, obor informační management.
- **Máte nějakou zkušenost s projektovým řízením a metodami pro časové plánování projektu?**
  Ano, nějaké má.
- **Znáte metodu kritické cesty?**
  Ano.
- **Znáte metodu Metra pothential method?**
  Ne.

Průběh testování Uživatel splnil všechny úkoly testovacího scénáře bez zaváhání. Při testování jsem si všiml, že pokud není splněna jedna z podmínek záporného potenciálu vazeb, tak je při smazání více entit najednou zobrazena ve stejném počtu varovná hláška o nesplněné podmínce.

Výstupní dotazník

- **Narazili jste v průběhu testování na nějaké problémy?**
  Při vytvoření vazby mezi dvěma kritickými činnostmi, mezi kterými vede jiná delší cesta, se vazba označila jako kritická.
- **Co se vám na aplikaci nelibí, co byste zlepšili?**
  Uživatel by si rád uložil rozpracovaný příklad v rámci aplikace, který by chtěl sdílet pomocí odkazu. Dále by se mu líbilo, kdyby se při vytváření vazby označila místa, na které lze vazbu připojit.
- **Co vás na aplikaci zaujalo?**
  Přehlednost a intuitivní ovládání aplikace. Dobře popsaný průběh výpočtu.
- **Pochopili jste problematiku, kterou se výuková aplikace snaží vysvětlit?**
  Uživatel uvedl, že problematiku pochopil.

5.4.4 Shrnutí

Při testování byly nalezeny následující chyby:

Vazba do neznáma Aplikace umožňovala vytvoření vazby bez cílové činnosti. O této vlastnosti jsem již věděl z implementace. Myslel jsem si však, že to bude užitečné, pokud budou činnosti daleko od sebe a uživatel bude muset posunout síťový diagram a až poté by napojil vazbu na danou činnost. Toto řešení se však ukázalo jako matoucí v případě, kdy uživatel nepřipojí správně vazbu. Dokud nepřesune činnost, tak vazba vypadá jako připojená. Tuto chybu jsem vyřešil automatickým smazáním vazby v případě, že není při vytvoření připojena na cílovou činnost.

Překrytí činností při vytváření Jeden z uživatelů si stěžoval na překrytí činností při jejich vytváření pomocí tlačítka z ovládacího panelu editoru. Nově vytvořené činnosti tímto způsobem se vytvářely uprostřed plochy editoru. Což znamenalo jejich překrytí, pokud jich uživatel vytvořil více a mezi tím je neposunul. Toto chování neberu jako chybu, ale na základě této připomínky jsem pozici nově vytvořených činností upravil o náhodný posun.

Špatné označení vazeb kritické cesty U vazby mezi kritickými činnostmi, mezi kterými vedla jiná delší cesta kritických činností, byla tato vazba označena rovněž jako kritická. Výpočet byl upraven tak, aby byla vazba označena za kritickou pouze v případě, pokud pomocí ni byla nastavena hodnota Early Finish cílového uzlu.

Dále z testování vzešlo několik návrhů na vylepšení aplikace:

- Možnost změnit barvu vizuálního editoru a jednotlivých činností.
- Vytvoření uživatelského účtu, v rámci kterého by bylo možné rozpracované příklady uložit.
- Možnost sdílet obsah editoru pomocí odkazu.
- Připojení vazby na činnost při uvolnění tlačítka myši v prostoru celé činnosti.

Kapitola 6

Dokumentace

6.1 Instalační příručka

Pro kompilaci musíte mít nainstalován správce balíčků `Yarn` [25] pro JavaScriptové projekty od firmy Facebook. Jedná se o open-source alternativu pro jiný správce balíčků jménem `NPM` [26]. `Yarn` využívá registrů `NPM`, je však lépe optimalizovaný. Pomocí tohoto správce balíčku nejdříve nainstalujete všechny knihovny aplikace v adresáři se zdrojovými kódy implementace:

```
yarn install
```

Dále je možné pokračovat jedním ze dvou příkazů:

yarn start Spustí aplikaci ve vývojovém módu spolu s lokálním serverem na adrese `http://localhost:3000`. Pokud provedete změny v kódu, stránka se automaticky znovu načte. V případě chyby se v prohlížeči zobrazí její popis včetně místa jejího vzniku.

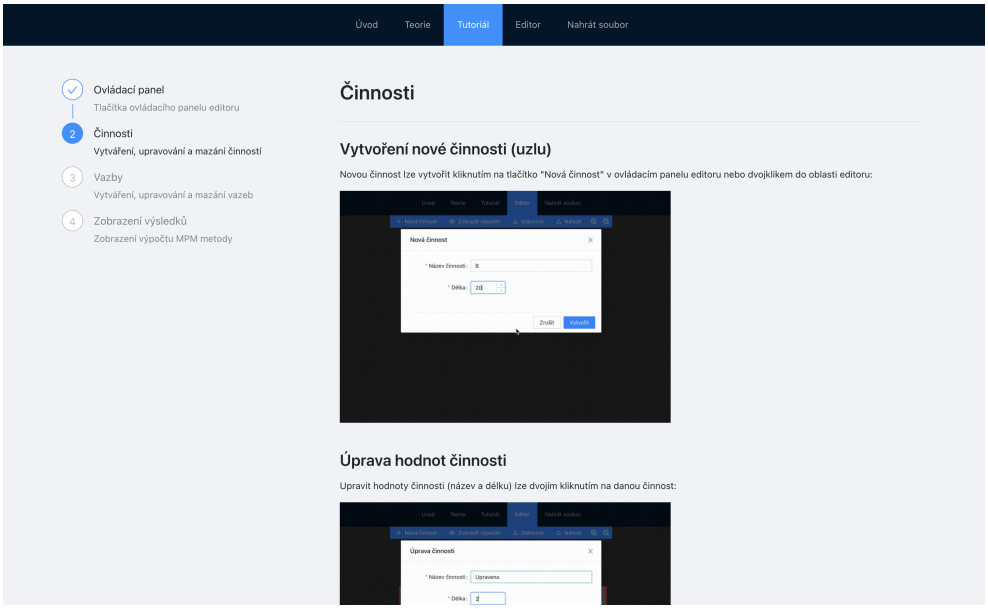
yarn build Vytvoří zdrojové kódy aplikace pro produkční prostředí do složky `./build`. V rámci sestavení produkční verze dochází k optimalizaci pro nejlepší výkon včetně sloučení a minifikace jednotlivých zdrojů.

Na přiloženém CD této práce se již nachází zkompilovaná verze aplikace v adresáři `build`. Pokud tedy nechcete upravovat zdrojové kódy aplikace, stačí přesunout obsah této složky na příslušný server nebo otevřít soubor `build/index.html` v prohlížeči.

6.2 Uživatelská příručka

Uživatelská příručka je řešena přímo v aplikaci formou tutoriálu popsaného v kapitole věnující se návrhu. Uživatel je proveden všemi funkcemi vizuálního

editoru pomocí animovaných obrázků. Jednotlivé funkce jsou rozděleny do logických skupin podle entit, ke kterým se vztahují. Na tuto stránku se uživatel může prokliknout ze všech stránek aplikace.



Obrázek 6.1: Stránka s tutoriálem

Kapitola 7

Zhodnocení aplikace a doporučení dalšího vývoje

V rámci implementace se podařilo splnit všechny požadavky na podporované procesy. Student si v aplikaci může nastudovat teorii k MPM metodě a nabyté znalosti následně otestovat ve vizuálním editoru. V tomto editoru jsou mu umožněny všechny akce spojené s vytvářením síťového diagramu pro simulaci metody MPM. Ovládání editoru je popsané v samostatné sekci nazývané jako tutoriál. Jednotlivé akce jako je přidávání činností nebo hran je znázorněno pomocí animovaných obrázků GIF, na kterých je názorně provedena daná akce. Uživatel si může aktuální stav editoru uložit do souboru a následně ho nahrát z jakékoliv stránky aplikace. Jelikož je MPM metoda pouze rozšířená metoda CPM o potenciály vazeb, tak lze vizuální editor využít i pro simulaci metody CPM ponecháním potenciálů vazeb na výchozí hodnotu.

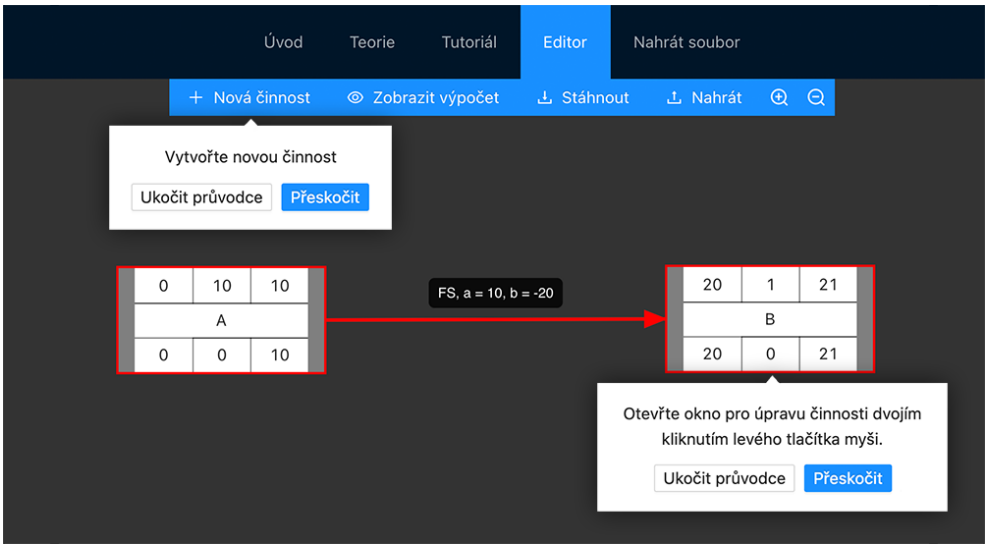
Celá aplikace byla implementována jako JavaScriptová webová stránka bez serverové části za využití vhodných knihoven, které značně urychlili její vývoj. V současné době podporuje základní procesy, které jsou definované v zadání této diplomové práce, ale již při implementaci vzešlo nespočet možností na její rozšíření:

Příklady s ověřením výsledku Aplikace by kromě vzorových příkladů, kdy je nahrán do editoru celý síťový diagram, mohla obsahovat také příklady, které by obsahovali pouze zadání. Podle zadání by měl student v editoru vytvořit síťový diagram. Tento diagram a výsledek metody by byl porovnán s výsledkem, který by uživatel neviděl a na základě toho by bylo vyhodnoceno jeho řešení.

Serverová část Jak již bylo zmíněno, tak pro aplikaci v rámci této diplomové práce nebyla implementována serverová část. Kvůli její absenci není možné například autentizovat uživatele nebo jednoduše přidávat nové vzorové příklady, či upravovat obsah teorie. Na druhou stranu je aplikace použitelná

bez připojení internetu z lokálního úložiště zařízení. Případná serverová část by mohla poskytovat přihlášení jak pro studenty tak učitele. Učitelé by mohli spravovat obsah teorie a přidávat nové vzorové příklady nebo různé úkoly pro studenty. K přihlášení do aplikace by mohla být v rámci integrace do systému univerzity použita její služba Single Sign-On, která umožňuje studentům a jejím pracovníkům přihlášení do všech aplikací univerzity jednotnými přihlašovacími údaji. Serverová část by měla být řešena jako API server, který by poskytoval klientské části aplikace REST rozhraní.

Interaktivní průvodce Uživateli by byl při první návštěvě vizuálního editoru spuštěn interaktivní průvodce vytváření síťového diagramu a zobrazení výpočtu metody formou oken zobrazujících se u dané akce. Tyto okna by se zobrazovala v závislosti na interakci uživatele s aplikací. Průvodce by uživatel mohl ukončit kliknutím na tlačítko „Ukončit průvodce“ nebo daný krok přeskočit kliknutím na tlačítko „Přeskočit“. K této funkcionalitě by mohla být využita komponenta `Popconfirm`, která je součástí Ant Design knihovny představené v kapitole 4.2.2.



Obrázek 7.1: Návrh interaktivního průvodce editorem

Znázornění typů vazeb Nehledě na typ vazby jsou všechny znázorněny od pravé strany uzlu k levé straně uzlu do kterého směřují, což značí vazbu typu *„finish to start“*. Zvolený typ vazby je pak umístěn do štítku, který se nachází uprostřed vazby viz. obrázek 7.1. Správně by se jednotlivé vazby měly v rámci uzlu napojovat na místo specifické pro jejich vlastnost. Levá strana činnosti by měla být brána jako její začátek a pravá strana jako její konec. Například pro vazbu typu *„start to start“* by měla šipka směřovat od levé

strany činnosti, ze které vystupuje do levé strany činnosti, do které vstupuje. V rámci dalšího vývoje mohla být implementována nová třída vykreslující vazby pro knihovnu React diagrams, která by tuto vlastnost implementovala. Automaticky by hrany zalamovala do pravého úhlu a uživatel by s nimi mohl posouvat, aby dokázal síťový diagram upravit při nepřehledném automatickém propojení. Knihovna v jejich výchozích třídách pro tvorbu hran podporuje přidávání takzvaných brakepointů, pomocí nichž se dá hrana v určitém bodě zalomit. Této funkcionality by mohlo být využito.

Závěr

V první části věnované teorii byl představen hlavní účel projektového řízení, jehož nedílnou součástí je časové plánování celého projektu. Tomuto plánování se věnují různé metody, mezi které patří i Metra potential method, pro jejíž výuku byla v rámci této diplomové práci vyvíjena výuková aplikace. Pro pochopení této a dalších metod časového plánování bylo v této kapitole definováno několik základních pojmů teorie grafů vztahujících se k síťovým diagramům. Následně byla popsána metoda kritické cesty, která je základem pro Metra potential method. Tato metoda rozšiřuje metodu kritické cesty o kladný a záporný potenciál, díky kterým umožňuje pozměnit časový plán projektu bez nutnosti úprav topologie síťového diagramu či hodnot jednotlivých činností. Jinak řečeno, lze v metodě kritické cesty ohodnotit pouze uzly (činnosti) diagramu. V metodě MPM lze díky potenciálům ohodnotit i hrany (vazby) mezi jednotlivými činnostmi.

V následující kapitole věnující se analýze a návrhu byly podrobně rozebrány jednotlivé procesy, které má aplikace podporovat. Na základě těchto procesů byly určeny funkční a nefunkční požadavky na aplikaci. Podle vydefinovaných procesů a funkčních požadavků na aplikaci byly určeny jednotlivé případy užití, které popisují interakci mezi uživatelem a aplikací. Tyto případy užití byly znázorněny pomocí grafického diagramu v jazyce UML. Následně byly hlavní průchody aplikací detailněji rozepsány ve scénářích. V těchto scénářích je po jednotlivých krocích rozepsána interakce mezi uživatelem a aplikací včetně alternativních událostí, které mohou nastat. Díky nim vznikla první představa o tom, jak bude aplikace konkrétně fungovat a jaké všechny obrazovky a prvky bude obsahovat. Podle jednotlivých případů užití bylo navrženo uživatelské rozhraní pomocí drátěných modelů (wireframů) jednotlivých obrazovek. Při návrhu byl kladen důraz na dodržení Nielsonovy heuristiky, jejíž body jsou v rámci této kapitoly popsány.

Aplikace byla implementována jako klientská aplikace spustitelná v prohlížeči bez nutnosti využití webového serveru. K implementaci byl z tohoto důvodu zvolen programovací jazyk JavaScript spolu s open-source nastav-

bou TypeScript, která tento programovací jazyk rozšiřuje mimo jiné o anotaci typů a typovou kontrolu, díky které lze jednodušeji předcházet chybám při špatné manipulaci s proměnnými a případné chyby snadněji opravovat. Při implementaci bylo využito několik knihoven pro urychlení implementace. Tou hlavní je framework React, který umožňuje efektivní vývoj JavaScriptových single-page aplikací. Pro udržení jednotného stylu aplikace bylo využito design systému Ant Design a jeho knihovny určené pro použitý framework React. V neposlední řade byla vyžita knihovna pro vytváření diagramů, ke které byly doprogramovány třídy tak, aby vyhovovala požadavkům na vizualizaci MPM metody. Z implementace vzešlo několik nápadů na další vývoj aplikace. Tyto náměty byly rozebrány v samostatné kapitole věnující se dalšímu vývoji.

Po dokončení vývoje aplikace bylo provedeno testování s uživateli podle jednoho testovacího scénáře, který pokrýval všechny funkcionality aplikace. Uživatelské testování bylo provedeno online za použití sdílené obrazovky. Při testování bylo nalezeno několik menších chyb, které jsem v rámci této diplomové práce opravil. Dalším výstupem testování byly návrhy od uživatelů na možné další rozšíření aplikace.

Výsledkem práce je výuková webová aplikace pro matematické modelování metodou MPM splňující všechny požadavky ze zadání diplomové práce. Věřím, že tato aplikace ulehčí studentům pochopit tuto metodu. Před psaním této diplomové práce jsem já sám neměl s metodami pro časové plánování žádné zkušenosti a v průběhu rešerše k teoretické části této práce mi trvalo nějaký čas metody pochopit. V průběhu implementace se však ukázalo, že jak metoda CPM, tak i metoda MPM jsou jednoduché matematické modely, které lze při grafickém znázornění snadno pochopit.

Literatura

[1] Ganttův diagram - Wikipedie [online]. [cit. 2020-10-02]. Dostupné z: https://cs.wikipedia.org/wiki/Gantt%C5%AFv_diagram

[2] Metra Potential Method - apppm [online]. [cit. 2020-10-07]. Dostupné z: http://apppm.man.dtu.dk/index.php/Metra_Potential_Method

[3] Institute, P. M.: *Project Management Body of Knowledge (PMBOK ® Guide)*. Project Management Institute, Inc., třetí vydání, ISBN 978-1930699458.

[4] Schwalbe, K.: *Řízení projektů v IT*. Brno: Comuter Press, a.s., 2007, čtvrté vydání, ISBN 978-80-251-1526-8.

[5] Milton D. Rosenau, J.: *Řízení projektů*. Praha: Computer Press, 2000, první vydání, ISBN 80-7226-218-1.

[6] Problém sedmi mostů města Královce. [cit. 2020-11-15]. Dostupné z: https://www.algoritmy.net/article/91/Problem-sedmi-mostu

[7] Teorie grafů - Základní pojmy [online]. [cit. 2020-11-20]. Dostupné z: http://books.fs.vsb.cz/SystAnal/texty/21.htm

[8] Work Breakdown Structure (WBS): The Ultimate Guide with Examples [online]. [cit. 2020-11-25]. Dostupné z: https://www.projectmanager.com/work-breakdown-structure

[9] GANTTŮV DIAGRAM [online]. [cit. 2020-12-06]. Dostupné z: http://www.eurohudebka.cz/dokumenty/ganttuv_diagram.pdf

[10] KUBÍKOVÁ, Daniela - Metódy plánovania projektov využívajúce sieťové grafy [online]. [cit. 2020-12-20]. Dostupné z: https://is.muni.cz/th/li7jf/dp.pdf

[11] Network planning by the extended metra potential method (EMPM) [online]. [cit. 2021-01-03]. Dostupné z: https://pure.tue.nl/ws/files/4368315/104181.pdf

[12] Metoda PERT (Program Evaluation and Review Technique) - ManagementMania.com [online]. [cit. 2020-01-23]. Dostupné z: https://managementmania.com/cs/metoda-pert

[13] GERT analysis - graphical evaluation and review technique [online]. [cit. 2021-02-02]. Dostupné z: https://www.pmi.org/learning/library/gert-graphical-evaluation-review-technique-5716

[14] Nielsen's Heuristic Evaluation - Vojta Svoboda Blog [online]. [cit. 2021-02-07]. Dostupné z: https://blog.vojtasvoboda.cz/nielsens-heuristic-evaluation

[15] JavaScript.com [online]. [cit. 2021-03-01]. Dostupné z: https://www.javascript.com/

[16] JavaScript ES6 [online]. [cit. 2021-03-01]. Dostupné z: https://www.w3schools.com/js/js_es6.asp

[17] React Native · Learn once, write anywhere [online]. [cit. 2021-03-01]. Dostupné z: https://reactnative.dev/

[18] NativeScript [online]. [cit. 2021-03-01]. Dostupné z: https://nativescript.org/

[19] TypeScript: Typed JavaScript at Any Scale. [online]. [cit. 2021-03-02]. Dostupné z: https://www.typescriptlang.org/

[20] React – A JavaScript library for building user interfaces [online]. [cit. 2021-03-15]. Dostupné z: https://reactjs.org/

[21] React Router: Declarative Routing for React.js [online]. [cit. 2021-03-20]. Dostupné z: https://reactrouter.com/

[22] History API - Web APIs — MDN [online]. [cit. 2021-03-20. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/History_API

[23] Ant Design of React - Ant Design [online]. [cit. 2021-04-06]. Dostupné z: https://ant.design/docs/react/introduce

[24] projectstorm/react-diagrams: a super simple, no-nonsense diagramming library written in react that just works [online]. [cit. 2021-04-07]. Dostupné z: https://github.com/projectstorm/react-diagrams

[25] Home — Yarn - Package Manager [online]. [cit. 2021-04-16]. Dostupné z: https://yarnpkg.com/

[26] npm [online]. [cit. 2021-04-16]. Dostupné z: https://www.npmjs.com/

Příloha **A**

Seznam použitých zkratek

CPM Critical path method

MPM Metra potential method

GERT Graphical Evaluation and Review Technique

PERT Program Evaluation and Review Technique

WBS Work breakdown structure

AON activity on node

AOA action on arrow

ES early start

EF early finish

LS late start

LF late finish

FS finish to start

SS start to start

FF finish to finish

SF start to finish

JSON JavaScript Object Notation

ES6 ECMAScript 2015

HTML Hypertext Markup Language

JSX JavaScript XML

SPA Single-page application

API Application Programming Interface

GIF Graphics Interchange Format

REST Representational state transfer

NPM Node Package Manager

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `build` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `DP_Vaclav_Dvorak_2021.pdf` ..... text práce ve formátu PDF