FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Vytvoření prototypu na zadávání speciálních algoritmů v prostředí Moodlu
Student:	Bc. Denis Drda
Vedoucí:	Ing. Petra Pavlíčková, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Manažerská informatika
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem práce je vytvoření prototypu se speciálními algoritmy na zlepšení tvorby testů matematických úloh v Moodlu.

Konkrétní dílčí cíle jsou následující:
1) Prozkoumejte možná řešení v rámci Moodlu.
2) Proveďte sběr požadavků na zadávání speciálních algoritmů v Moodlu.
3) Analyzujte možnosti tvorby modulů (pluginů) do Moodlu
4) Navrhněte prototyp.
5) Vytvořte prototyp do Moodlu.
6) Vypracujte uživatelskou příručku.
7) Vytvořený prototyp otestujte a zdokumentujte.
8) Řešení zhodnoťte a doporučte další postup.

Elektronicky schválil/a Ing. David Buchtela, Ph.D. dne 29. října 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Vytvoření prototypu na zadávání speciálních algoritmů v prostředí Moodlu

Denis Drda, Bc.

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

30. dubna 2022

Poděkování

Rád bych poděkoval vedoucí práce paní doktorce Petře Pavlíčkové, za podporu, vstřícný přístup a cenné rady během přípravy práce. Rovněž bych poděkoval celé své rodině a přátelům za podporu při jejím psaní mé práce a podporu celého období mého 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 předpisu. 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 využít. Tyto osoby jsou oprávněny Dílo využít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelu (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 30. dubna 2022 . . . . . .

Odkaz na tuto práci

Drda, Denis. *Vytvoření prototypu na zadávání speciálních algoritmů v prostředí Moodlu*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Abstrakt

Práce se zaměřuje na vytvoření prototypu na zadávání speciálních algoritmů v prostředí Moodlu. Kromě toho pojednává o souvisejících pojmu, informacích důležitých nejen k vývoji, ale i pro analýzu a návrh.

Teoretická část práce se zaměřuje na definování podstatných pojmů souvisejících s prací. Jako je informace a použití úloh z Lineárního programování, podrobněji Simplexového algoritmu. Pojednává o elektronickém vzdělání pomocí systému Moodle. Spolu s předchozím cílem je důkladná analýza běžně dostupného Moodlu i vývojového prostředí systému. Kapitola se bude věnovat i programovacím pravidlům a možnostech ke sdílení vytvořeného modulu.

Praktická část v úvodu se bude zabývat nainstalováním vývojového prostředí Moodle pro tvorbu testů. Konkrétně vytvoření testové úlohy na testování Simplexové úlohy v základní instalaci bez doplňků. Poté následuje hlavní část práce o vývoji nového prototypu na zadávání speciálních algoritmů, jako jsou potřebné oblasti analýza řešení, návrh a následná implementace prototypu. Dále bude vypracovaná uživatelská příručka o instalaci a ovládání uživatelského rozhraní. V práci následuje otestování a zdokumentování vytvořeného modulu. Celá práce v poslední řade bude zakončena zhodnocením a doporučením dalšího postupu pro vytvořený prototyp pluginu.

Klíčová slova Lineární programování, Simplexový algoritmus, e-learnig, Moodle, LMS, vývoj Moodlu, testová úloha, test

Abstract

The work focuses on creating a prototype for entering special algorithms in the Moodle environment. In addition, it discusses related concepts, information important not only for development but also for analysis and design.

The theoretical part of the thesis focuses on defining essential terms related to work. Such as information and application tasks from Linear Programming, in more detail the Simplex algorithm. It deals with e-learning using the Moodle system. Together with the previous goal, there is a thorough analysis of the commonly available Moodle and the development environment of the system. The chapter will also deal with programming rules and options for sharing the created module.

The practical part in the introduction will deal with the installation of the Moodle development environment for creating tests. Specifically, creating a test task to test a Simplex task in a base installation without add-ons. This is followed by the main part of the work on the development of a new prototype for entering special algorithms, such as the necessary areas of solution analysis, design and subsequent implementation of the prototype. Furthermore, a user manual on installation and user interface control will be developed. The work is followed by testing and documentation of the created module. The last work in the last row will end with the evaluation and recommendation of further steps for the created prototype of the plugin.

Keywords Linear programming, Simplex algorithm, e-learning, Moodle, LMS, Moodle development, test task, test

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V dnešní internetové a počítačové době se vzdělávání umožňuje čím dál víc na dálku. Zmíněnou skutečnost především urychlila Koronavirová krize (pandemie Covid), při které byli omezeny osobní kontakty mezi lidmi. To platilo i v oblastech vzdělávání a školství, byli zavřené školy. Vyučovalo se na dálku pomocí internetu. Vzdáleně, přes internet se zkoušelo a testovalo. Tím se podle mého názoru velmi urychlila elektronizace školství a elektronické vzdělávání. Což bylo pro mnohé nemyslitelné a zdlouhavé. Díky tomu se výuka a testování přesunovalo do online prostoru. Součástí dnešního života už je normální online schůzka na vyučovací hodinu, či zkoušení.

Záměrem mojí diplomové práce je nejen využít zmíněného rozvoje elektronického vzdělávání, ale možnost přispět i něčem dalšímu užitečnému. Jedná se zefektivnění zadávání testových úloh na Lineární programování, které v dnešní době de facto neexistuje. Prototyp modulu by měl umožnit v systému Moodle testování speciálních algoritmů a následně automaticky vyhodnocovat výpočty, aby student ihned věděl výsledek svého testu.

Jedná se o využití jednoho z nejpopulárnějších elektronických nástrojů ke vzdělávání. Zkratka Moodle vychází ze slov: modulární objektově orientované výukové prostředí. Tím lze snadno vyvíjet a upravovat nové potřeby, pomocí pluginů. Plugin je ekvivalent slova modul. Celý systém je vyvíjen a poskytován zdarma. Vývoj je pod volnou licencí GNU. To znamená, že autorské práva nabízí uživatelům velkou svobodu nejen v úpravách v souladu s licenčním podmínkami.

Práce je pro přehlednost rozdělena do několika částí. První část hovoří o cílech práce. Teoretická část pojednává základních pojmech potřebných k práci. Konkrétně se jedná o specifikaci matematických speciálních algoritmů, včetně průzkumu a podrobného rozboru příkladu Simplexového algoritmu. V poslední řadě se věnuje informacím, používání sytému Moodle. Nechybí, ani informace o možných testovacích prostředcích s doporučením vybrané testovací úlohy na zmíněný algoritmus. Práce se bude zabývat základními informace o modulech, včetně jejich instalace. Ale i vývojového prostředí Moodle sloužící k rozvoji a implementaci různé modularity. Na konci této kapitoly budou pravidla pro programování a pro sdílení vytvořeného pluginu.

Následující kapitola se bude týkat praktické části, hlavně se zaměřením na vytvoření prototypu na zadávání speciálních algoritmů v prostředí Moodlu. Úvod se bude věnovat možnostem nainstalování Moodlu. Další kapitola bude pojednávat o vytvoření testu na konkrétní speciální algoritmus v základní instalaci Moodle. Poté se v práci budu věnovat nejdůležitější části vývoje prototypu. Kde nebude chybět analýza, ani návrh pluginu a nakonec i samotná implementace. Následuje uživatelská příručka, nejen o instalaci, ale i uživatelském rozhraní. Z pohledu učitele a studenta. Poslední podkapitola bude pojednávat o testování a zdokumentování vytvořeného modulu. Závěrem práce vyhodnotím celý naimplementovaný prototyp a následně doporučím další postup pro úspěšný vývoj dokonalého pluginu.

Kapitola 1

Cíl práce

Hlavním cílem práce je vytvoření prototypu na zadávání speciálních algoritmů v prostředí Moodlu, konkretně naimplementovaní nové testové úlohy podle požadavků na vkládání zadání úlohy z Lineárního programovaní. Následuje rešerše o elektronickém vzdělání pomocí systému Moodle. Z předchozího cíle vyplývá důkladná analýza Moodlu. Dalším cílem je průzkum možných řešení testování v systému Moodle, ale i vývojového prostředí systému. Následuje vypracovaní uživatelské příručky o instalaci a ovládání uživatelského rozhraní. Mezi poslední úkole patří otestování a zdokumentování vytvořeného modulu. Závěrem zhodnotím a doporučím další postupu pro vytvořený prototyp pluginu na zadávání speciálních algoritmů.

Kapitola 2

Teoretická část

Nejprve začnu s definování speciálních algoritmů, kterému se budu věnovat v práci. Jako zmíněný problém jsem si vybral úlohy z Lineární programování. Podrobněji se zaměřím na Simplexový algoritmus zabývajícím se nalezení optimálního řešení. Dále se zaměřím i na řešení Lineárního optimalizačního modelu. Ukážu i řešení příkladu.

Stručně definuji pojem a využití elektronického vzdělání. V další části se budu věnovat vzdělávacímu systému Moodle. Kde nebudou chybět nejdůležitější informace, uspořádání a možnosti v ovládání systému. Podrobně se zaměřím na druhy testování, fungování a jejich tvorby v Moodlu. Následně vyberu a provedu rešerši nejvhodnější testové úlohy k využití k testování speciálních algoritmů. Další kapitolu věnuji základním informacím o vytvářením testu v systému Moodlu.

Nebude chybět kapitola o zásuvných modulech v systému Moodle i o jejich instalaci a odstranění. V další části se zaměřím na možný vývoj a rozšíření celého Moodle. Provedu rešerši architektury a rozhraní systému. Uvedu základní informace o programování a požadavků na vývoj Moodlu. Nakonec představím pravidla pro sdílení a přispívání pluginu pro komunitu Moodle.

2.1 Lineární programování

Lineárního programování řeší především nalezení minima či maxima lineární funkce při několika proměnných v soustavě lineárních rovnic. V rámci řešení problémů je zapotřebí nalézt nejlepší možné řešení, to má na starost optimalizační model. Celé řešení lineárních modelů je prováděno univerzální matematickou Simplexovou metodou. Zmíněná metoda je založena na Jordanově eliminační metodě. Pro lepší představu budu řešit příklad simplexového algoritmu v jednotlivých krocích.

V následujících podkapitolách budu velmi často citovat z knihy Ekonomicko-matematické metody[1], ze skript Základní metody operační analýzy[2] a z knihy Lineární programování[3] pro správnost a odbornost použitých výrazů.

2.1.1 Lineární programovací model

Optimalizační model, který patří mezi hlavní cíle lineárního programování. Jehož úkol spočívá v nalezení optimálního rozsahu v závislosti na nárůstu extrému funkcí, podle zadaných kritérií. Lineární rovnice a nerovnice, včetně kritérií lineárních funkcí jsou brány jako omezující podmínky v lineárním modelu.

„Cíl
Nalezení řešení x, které splňuje všechny omezující podmínky a účelová funkce v něm nabývá požadované optimální hodnoty.“[1, strana 16]

2.1.1.1 Popis modelu lineárního programování

Nejprve začnu s definováním požadovaných jednotlivých procesů s vyjádřených v jednotkách vhodných pro náš příklad. Potřebuji rovněž vyjádřit extrém účelové funkce, omezení pro hledání řešení, včetně určený dalších hodnot. Jedná se technickoekonomické koeficienty, které jsou nezbytné pro popis vlivu jednotlivých procesů v určité podmínce. Např. jednotlivé kapacity, požadavky, bilanční rovnováhy, atd. Zapisují se pomocí extrémů, proměnných, omezujících podmínek, včetně podmínek nezápornosti. Zmíněným termínům se budu věnovat v tomto odstavci.

V závěru popíši kritéria rozhodování pomocí cenových koeficientů. Zmíněné koeficienty vyjadřují zda jsou jednotlivé procesy výhodné či nevýhodné, v rámci splnitelnosti zadaných omezení. Nelze obecně říci, že nevýhodný cenový proces je špatný. Je možné, že proces s menší výhodností lépe přispěje k hodnotě kritérií, než proces s výhodnějším koeficientem.

„Nalézt extrém účelové funkce

$$z(x) = c^T x \rightarrow MIN$$

Omezující podmínky

$$\begin{matrix} \leq \\ AX = b \\ \geq \end{matrix}$$

Podmínky nezápornosti

$$x \geq 0$$

kde $x = (x_1, ..., x_n)^T$ *je vektor proměnných,*

$z(x) = c^T x = \sum_{j=1}^{n} c_j x_j$ *je účelová funkce,*

$c^T = (c_1, ..., c_n$ *je vektor cen nebo sazeb proměnných,*

$A = (A_{ij}$ *je matice technickoekonomických koeficientů,*

$b = (b_1, ..., b_n)^T$ *je vektor pravých stran soustavy omezujících podmínek.*

$x \geq 0$ *jsou podmínky nezápornosti,* $x_j \geq 0, j = 1, \ldots n.$

Soustava omezujících podmínek

$$a^T{}_1 x = \sum_{j=1}^{n} a_{1j} x_j \leq b_1$$

$$\vdots$$

$$a^T{}_m x = \sum_{j=m}^{n} a_{mj} x_j \leq b_m$$ “[2, strana 57]

Uvedené podmínky nezápornosti hodnot proměnných jsou důležité především k Simplexovému algoritmu.

Bez těchto podmínek by nebylo možné dokázat jednotlivé kroky.

„Prvky lineárního optimalizačního modelu:

- *vektor proměnných* $x = (x_1, ..., x_n)^T \in R^n$*, který popisuje jednotlivé složky hledaného rozhodnutí*
- *omezující podmínky* $Ax \leq b, i = 1, ..., m$*, které popisují reálná omezení hledaných rozhodnutí*
- *účelová nebo kriteriální funkce* $z(x) = c^T x$*, která popisuje cíl, kritérium hledaného rozhodnutí*“[2, strana 57]

2.1.1.2 Formulace lineárního programování

Pro formulaci lineární optimalizační úlohy neexistuje univerzální návod. Matematický výpočet je jednoduchý. Obtížnost výpočtu spočívá v aplikaci. Model LP se využívá zejména v rozhodovacích situacích, kdy je možno realizovat větší počet činností (procesů) v různých kombinacích a je třeba stanovit podle určitého hlediska (například maximalizace zisku) optimální kombinaci těchto činností.[1]

Nejdůležitější je v rámci zkoumaného systémů pečlivě definovat prvky. Jsou obsaženy v modelu za pomocí lineárních rovnic a nerovnic. Rovnice a nerovnice v modelu popisují vztahy prvků definovaného systému a koeficienty proměnných vyjadřují vlastnosti jednotlivých prvků. Lineární účelová funkce popisuje výsledek chování systému, kritérium systému a její koeficienty z tohoto hlediska charakterizují jednotlivé prvky.[2]

Během tvorby omezujících podmínek modelu je potřebné nezapomenout na zachování sčitatelnosti stejných jednotek u všech členů, popř. převést na stejné jednotky. Tato podmínka slouží jako základní myšlenka pro sestavení správné formulace příkladu.

Nutné je nezapomenout na vlastnosti lineární úlohy při čerpání informací ze zadání. Především se jedná o vlastnosti, jako jsou: Linearita, statičnost a determinističnost. Zmíněné vlastnosti dokládají, že nalezené optimální řešení je silně závislé na malých změnách hodnot. Optimální řešení je možné pouze se stejnými výchozími či velmi blízkými parametry hodnot.

2.1.2 Základní úlohy řešitelné za pomocí lineárního optimalizačního modelu

Pro řešení existuje řada metod. Mezi jednodušší metody patří použití modelu ve více variantách s různými vstupními parametry. Zmíněné lineární modely, lze využít pro různorodá řešení problémů.

Optimalizace výrobní struktury cílem tohoto modelu je nalézt optimální rozsahy výrobních procesů v rámci daných výrobních kapacit.[2]

Alokační problémy do této skupiny patří všechny problémy, ve kterých jde o rozdělení zdrojů na pořízení určitých objektů, např. optimalizace portfolia, optimalizace formy reklamy, volba technologií, nebo o přiřazení zdrojů, např. pracovníků s různou kvalifikací na zajištění různých činností.[2]

Směšovací problémy úlohy se zabývají nalezením optimálních množství jednotlivých složek směsi tak, aby byly zajištěny všechny vlastnosti výsledné směsi.[2]

Problémy dělení materiálů úlohy jsou často nazývány úlohami o řezných plánech, jde o optimální volbu způsobu dělení materiálu, aby byla zajištěna požadovaná množství jednotlivých jeho částí.[2]

Distribuční problémy jde o optimalizaci distribuce zboží mezi dodavateli a odběrateli. Tyto úlohy mají některé specifické vlastnosti[2]

2.1.3 Omezující podmínky

V lineárním optimalizačním modelu jsou dvě vazby podmínek. Jedná se o vnější a vnitřní vazby systému.

Vnější vazby v systému

Kapacitní *„modelují nemožnost vyčerpání většího množství zdroje, než je v systému k dispozici; zapisujeme*

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i$$

kde a_{ij} představuje množství vyčerpaného i-tého zdroje jednou jednotkovou j-tého procesu a b_i je maximální dostupné množství tohoto zdroje.“[1, strana 16]

Konkrétně se jedná o: disponibilní množství výrobních prostředků.[2]

Požadavkové *„modelují potřebu zajistit alespoň dané produkce; zapisujeme*

$$\sum_{j=1}^{n} a_{ij}x_j \geq b_i$$

kde a_{ij} je množství produkce i-tého typu vyrobeného j-tým procesem a b_i vyjadřuje minimální množství produkce i, které je třeba vyrobit“[1, strana 17]

Konkrétně se jedná o: výrobní úkoly, dolní omezení zisku, atd.[2]

Určené *„modelují nutnost dosáhnout stanoveného výstupu, tedy*

$$\sum_{j=1}^{n} a_{ij}x_j = b_i$$

kde a_{ij} je množství produkce i-tého typu vyrobeného j-tým procesem a b_i vyjadřuje přesně stanovené množství produkce i, které je potřeba vyrobit.“[1, strana 17]

Konkrétně se jedná o: používají jen velmi omezeně a vyjadřují požadavek přesně dané hodnoty[2]

Vnitřní vazby v systému

Bilanční *„modelují vztah mezi produkcí P a s spotřebou S,*

$$P = \sum_{j \in P} a_{ij}x_j = \sum_{k \in S} a_{ik}x_k = S$$

kde P je množina indexů produkce a S je množina indexů spotřeby.“[1, strana 17]

Konkrétně se jedná o: rozdělení produkce

Poměrové *„modelují požadavek, aby uvažované komponenty byly v určitém poměru K, například*

$$\frac{a_{ij}x_j}{a_{ik}x_k} = K \text{ , po úpravě } a_{ij}x_j - Ka_{ik}x_k = 0\text{“}[1, strana17]$$

2.1.4 Účelová (kriteriální) funkce

Jedná se o zastupování cíle pro řešení problému, protože hodnotí kvalitu přístupných kombinací procesu. *„Kvalita každého procesu je ohodnocena tzv. cenovým koeficientem c_j, který vyjadřuje příspěvek jedné jednotky tohoto procesu k celkové hodnotě sledovaného kritéria.*

$$z = \sum_{j=1}^{n} c_j x_j \rightarrow MAX\text{“[1, strana 18]}$$

2.1.5 Podmínky nezápornosti

Hlavní význam nezáporných proměnných vyplývá z výpočetních důvodů. *„Matematicky je možné zapsat model LP více způsoby. Relativně úsporně lze model zapsat v maticové formě následujícím způsobem:*

$$Ax \leq b$$

$$x \geq 0$$

$$z = c^T x \rightarrow MAX\text{“[1, strana 18]}$$

2.1.6 Shrnutí základní teorie modelu lineárního programování

- Stanovení neznámých, jako jsou x1, x2, atd.
- Vymezení omezující podmínky:
  - Kapacitní $\leq$
  - Požadavkové $\geq$
  - Určující $=$
- Účelová funkce:
  - Z výsledku proměnné peníze můžu mít zisk, resp. ztrátu
  - Minilizační nebo maximalizační
- Podmínky nezápornosti:

  Všechny neznámé jsou ≥ 0
- Model lineárního programování
  - Prostor řešení (proměnné)
  - Prostor požadavků (omezující podmínky)
  - Simplexový algoritmus (výpočetní část)

2.2 Simplexový algoritmus

Je metoda k nalezení optimálního řešení, která je založena na početním řešení lineárních rovnic a nerovnic. Zároveň respektuje dané omezující podmínky. Podle[2] je metoda braná jako univerzální metoda pro řešení úloh lineárního programování. Tento postup hledání řešení využívá Jordanovou eliminační metodu doplněnou navíc o dvě kritéria, které slouží k nalezení optimálního řešení.

„Používá následující postup. Začne zkoumat nějaké bázické řešení, zda-li je optimální. Jestliže ano, tak skončí, když ne, nalezne nové bázické řešení, kterému odpovídá menší, nebo v krajním případě stejná hodnota účelové funkce, než v předchozím případě, a postup se zopakuje. Tím, že se metoda nevrací k horšímu řešením než je právě zkoumané řešení, se počet zkoumaných řešení podstatně zredukuje. Přechod od zkoumaného řešení k novému se děje pomocí výměny jednoho bázického vektoru.“[3, strana 32]

Princip zmíněné metody, včetně všech potřebných definic a výrazů představím v této podkapitole.

2.2.1 Soustava lineárních rovnic

Zmíněná soustava lineárních rovnic je vhodná k řešení problémů v matematice. Především část teorie lineárních rovnic má podrobně rozpracované metody, včetně jejich možných řešení. Řešení lineárních optimalizačních úloh probíhá pomocí Jordanovy eliminační metody s omezením některých eliminačních úprav. Pro použití soustavy lineárních rovnic je potřebné nejprve definovat, zda se jedná se o soustavu m lineárních rovnic o n neznámých proměnných.

„Soustava m lineárních rovnic o n proměnných může být zapsaná v maticovém tvaru:

$$Ax = b$$

kde $x = (x_1, ..., x_n)^T$ *je vektor proměnných*

$$A = (a_1, a_2, ..., a_n) = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ \vdots & & & \vdots \\ a_{m1} & am2 & ... & a_{mn} \end{pmatrix}$$

je matice koeficientů proměnných

$b = (b_1, ..., b_m)^T$ *je vektor pravých stran* “[2, strana 66]

Matice A je matice soustavy a matice $A|b$ značí rozšířenou matici soustavy lineárních rovnic.

Podle studijního textu k předmětu BI-LIN[4], lze definovat jako systém rovnic: *„Nechť* n *a* m *jsou přirozená čísla a pro všechna* $i \in \{1, \ldots, m\}$ *a* $j \in$

$\{1, \ldots, n\}$ *platí, že* $a_{ij} \in \mathbb{R}$ *a* $b_i \in \mathbb{R}$. *Sytém rovnic*

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_2 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ \vdots & + & \vdots & + & \ddots & + & \vdots & = & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{mn}x_n & = & b_m \end{array}$$

nazýváme ***soustavou*** *m* ***lineárních rovnic o*** *n* ***neznámých*** $x_1, \ldots, x_n \in \mathbb{R}$, *číslu* a_{ij}, *říkáme j-****tý koeficient*** *i-****té rovnice****."*[4, strana 8]

Obě definice jsou ekvivalentní. První definice[2], říká, že lineární rovnice zapíši do matice. Druhá definice[4], se znázorňuje systém rovnic. To znamená lineární rovnice zapsané pod sebe.

Soustavou lineárních rovnic lze získat kanonický (bázický) tvar, pokud její matice obsahuje úplnou jednotkovou matici řádků jako svou submatici.

„Po přerovnání je možné zapsat jako $A = (\widetilde{A}|E)$

kde A *je matice soustavy*

$\widetilde{A}$ *je část matice soustavy pro vyřazení sloupců jednotkových vektorů*

E *je jednotková matice řádu* m

Frobeniova věta:
Soustava lineárních rovnic je řešitelná, je-li hodnost matice soustavy rovna hodnosti rozšířené matice soustavy.
Je-li tato hodnost rovna počtu proměnných, je řešení jediné, je-li počet proměnných větší, je řešení nekonečně mnoho."[2, strana 66]

2.2.1.1 Jordanova eliminační metoda

Jedná se o metodu řešení lineárních rovnic. Jejíž hlavním úkolem je vytvoření ekvivalentní soustavy z původní soustavy rovnic s jednotkovou maticí. Celá Jordanova metodou vede k eliminaci lineárních rovnic do kanonického tvaru.

Metoda spočívá v převodu matice na redukovaný stupňovitý tvar. Princip použité metody spočívá v úpravě všech prvků mimo diagonálu. Při převodu na jednotkovou matici zároveň počítám inverzí matici.

2.2.1.2 Bázické, nebázické a parametrické řešení

Bázické proměnné jsou koeficienty kanonických proměnných vytvářející jednotkovou matici. Nebázickými proměnnými nazvu všechny ostatní proměnné. Vektory koeficientů u bazických proměnných v původním tvaru soustavy rovnic tvoří matici B, která obsahuje bázické vektory vektorového prostoru soustavy rovnic. Tyto vektory tvoří čtvercovou regulární soustavu lineárních rovnic, jejíž řešení je ekvivalentní s řešením původní soustavy, jsou-li nebázické proměnné rovny

nule. Z uvedeného důvodu každé bázické řešení obsahuje maximálně m nenulových složek. Bázickým (základním) řešením soustavy lineárních rovnic nazýváme takový vektor x, jehož nenulové složky odpovídají bázickým vektorů.[2]

Parametrické řešení dostaneme z hodnot vyjádřených nebázickými proměnnými, které budu považovat za parametry. Jedná se o soustavu lineárních rovnic vyjádřenou hodnotami bázických proměnných pomocí parametrů mínus hodnot nebázických proměnných.

$$x_i = b_i - \sum_{k=1}^{n-m} a_{ik}x_k\text{[2]}$$

Když dosadíme určité hodnoty za nebázické proměnné, dostanu konkrétní hodnoty i pro bázické proměnné. Zmíněné řešení se nazývá nebázické nebo nezákladní řešení. Nutno podoktnout, že parametrické nebo nezákladní řešení není určeno jednoznačně, záleží na konkrétní eliminaci.

2.2.1.3 Matice transformace

Pro použití pojmu matice transformace je potřebné nejprve definovat inverzní matici a následně matici transformace. Inverzní matice k dané regulární čtvercové matici B označíme B^{-1}, platí pro ně
$B.B^{-1} = B^{-1}.B = E$[2]

Matici B^{-1} nazveme matice transformace. *„Postup Jordanovy eliminační metody totiž lze také charakterizovat jako vynásobení celé soustavy lineárních rovnic inverzní maticí báze řešení. Takže každý vektor a_j vyjádřený v původní bázi je transformován na vektor α_j vyjádřený v bázi B a vektor pravých stran b na vektor β.“*[2, strana 68]

Při postupu Jordanovy eliminace provedu od výchozího tvaru rozšířené matice soustavy k transformovanému tvaru. Po provedení transformace jsou obě soustavy rovnic ekvivalentní.
$(A|E|b) \approx (\widetilde{A}|E|B^{-1}|\beta)$[2]
Po aplikování Jordanovy metody dostanu na místě jednotkové matice E matici inverzní B^{-1} k bázickým vektorům B.

2.2.2 Řešení lineárního optimalizačního modelu

Princip řešení lineárního optimalizačního modelu se zakládá na nalezení vhodného bázického řešení soustavy lineárních rovnic, včetně odpovídajících omezujících podmínek. Bázické řešení vypočtené pomocí Jordanovy eliminační metody, u kterého je potřebné zjistit, zda lze nalézt jiné lepší řešení s hodnotu účelové funkce. Pokud ano, je potřebné pomocí eliminace nalézt jinou vhodnou bazi.

Zmíněný postup je podrobně popsán v Simplexovém algoritmu.

V následujících odstavcích představím základní kroky a poté zformuluji celý algoritmus.

2.2.2.1 Kritérium optimality řešení

Kritérium optimality řeší, zda lze nalézt k řešení x^p soustavy omezujících podmínek. Jedná se o nalezení nejlepší hodnoty kritéria účelové funkce řešení. Poté nastanou dvě možnosti. První neexistence řešení, to znamená, že řešení x^p je optimálním řešením lineárního optimalizačního modelu. V druhé možnosti řešení existuje a zmíněné řešení x^p nemůže být optimální.

Nejprve začnu určením hodnot účelové funkce jako vztah:

$$„z(x^p) = c^T x^p = c_N^T x_N^p + c_B^T x_B^p = c_N^T 0 + c_B^T x_B^p = c_B^T x_B^p = c_B^T \beta$$

Jiné řešení soustavy omezujících podmínek je možno např. zapsat jako parametrické řešení soustavy vycházející z bázického řešení x^p, kde nebázická proměnná x_k nenulová, tedy pro x^{p+1} bude platit

$$x^{p+1} = (x_N^{p+1}, x_B^{p+1})^T = (0, x_k, 0, \beta - \alpha x_k)^T$$

a hodnota účelové funkce pro nové řešení bude

$$z(x^{p+1}) = c^T x^{p+1} = c_N^T x_N^{p+1} + c_B^T x_B^{p+1} = c_N^T (0, x_k, 0)^T + c_B^T (\beta - \alpha_k x_k)^T =$$

$$= c_k x_k + c_B^T (\beta - \alpha_k x_k)$$

Změna hodnoty účelové funkce pro řešení x^p a x^{p+1} je tedy rovná

$$z(x^{p+1}) - z(x^p) = c_k x_k + c_B^T (\beta - \alpha_k x_k) - c_B^T \beta =$$

$$= -(c_B^T \alpha_k - c_k) x_k = -(z_k - c_k) x_k$$

Výraz

$$z_k = c_B^T \alpha_k = \sum\nolimits_{i=1}^{m} c_i \alpha_{ik}$$

vyjadřuje cenu ekvivalentní kombinace bázických procesů.“[2, strana 69]

Cenový koeficient c_k proměnné x_k vyjadřuje přínos jednotkové úrovně k-tého procesu. Lineární kombinace z_k přestavuje přínos lineární kombinace bázických procesů. Porovnání těchto hodnot slouží k určení optimálního řešení.

„Nejjednodušší forma kritérií optimality řešení říká, že v danném kroku vypočítáme pro nebázické proměnné všechny rozdíly $(z_k - c_k)$ a z nich vybereme nejmenší, resp. Největší, podle směru optimalizace – maximalizace, resp. Minimalizace. Pokud nejmenší hodnota $(z_s - c_s)_{min}$ je menší než nula, resp. Největší hodnota $(z_s - c_s)_{max}$ je větší než nula, nalezení řešení není optimální a hodnota proměnné x_s by měla být nenulová, proměnná x_s by měla být zařazena do báze.“[2, strana 70]

Kritérium optimality pro maximalizační úlohu je jako rozdíl:

$$(z_s - c_s)_{min} = min_{s \in N}(z_k - c_k) = min_{s \in N}(c_B^T \alpha_k - c_k) \text{ [2]}$$

Kritérium optimality pro minimalizační úlohu je jako rozdíl:

$$(z_s - c_s)_{max} = max_{s \in N}(z_k - c_k) = max_{s \in N}(c_B^T \alpha_k - c_k) \text{ [2]}$$

Pro hodnoty testu optimality mohou nastat následující situace:

- při záporné hodnotě se hodnota účelové funkce se zvýší,
- při kladné hodnotě se hodnota účelové funkce se sníží,
- při nulové hodnotě se hodnota účelové funkce se nezmění.

2.2.2.2 Kritérium přípustnosti řešení

Pro kritérium přípustnosti řešení je zapotřebí nalezení bázické řešení nezáporné, to znamená $\beta_i > 0$ pro $i = i, \ldots, m$. Vyberu některou z nebázických proměnných $x_k, k \in N$ zařadím na některé místo z bázických proměnných $x_r, r \in B$, následně jí vyloučím z řešení. Zároveň potřebuji, aby řešení v bázi B_1 bylo stále nezáporné a hodnota účelové funkce ještě lepší.

Cílem je nalezení konečně maximální hodnoty účelové funkce za podmínky konečné proměnné hodnoty x_k. Existující také koeficient $\alpha_{ik} > 0$, musí být alespoň jeden. V případě, že nebude existovat ani jeden, tak nelze získat konečnou hodnotu účelové funkce.

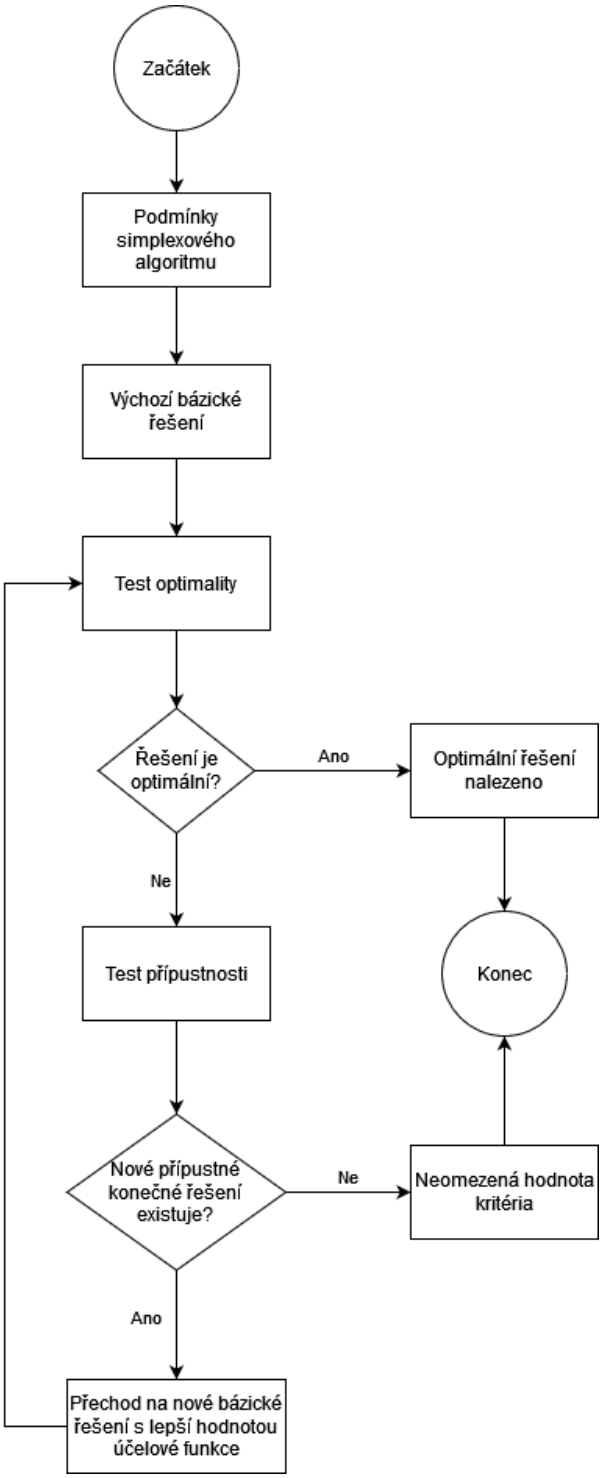
Kritérium přípustnosti budu definovat jako podíl:

$$\Omega^{(k)}{}_{min} = min_{\alpha_{ik} > 0 \frac{\beta_i}{\alpha_{ik}}} = \frac{\beta_r}{\alpha_{rk}} \text{ [2]}$$

Po dosazení do vektoru parametrického řešení lineárních rovnic s parametrem x_k dostanu řešení:

$$x = \begin{pmatrix} 0 \\ x_k \\ 0 \\ \beta - \alpha x_k \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{\beta_r}{\alpha_{rk}} \\ 0 \\ \beta - \alpha x_k \\ 0 \\ \beta - \alpha x_k \end{pmatrix} \text{ [2]}$$

Nově nalezené řešení splňuje omezující podmínky, neboli přípustnost. Musí být nezáporné, ale i bázické. Zmíněné řešení je prováděno jedním krokem Jordanovy eliminační metody. *„Kritérium přípustnosti řešení při změně báze říká, že v daném kroku ve sloupci k, který je vektorem transformovaných koeficientů zařazovaného proměnné x_k vyhledáme všechny kladné koeficienty α_{ik} a určíme podíly $\Omega^{(k)}$. Z těchto podílů vybereme nejmenší, tj. určíme $\Omega^{(k)}{}_{min}$, podle něhož nalezneme r-tý řádek, jemuž přísluší bázická proměnná x_r, kterou vyloučíme z báze.“*[2]



Obrázek 2.1: Schéma simplexového algoritmu
Zdroj: tvorba Bc. Denise Drdy

Tabulka 2.1: Jednotková submatice v matici soustavy omezujících podmínek[1]

x_1	x_2	d_1	d_2	p_2	p_3	b
a_{11}	a_{12}	**1**	0	**0**	**0**	b_1
a_{21}	a_{22}	**0**	-1	**1**	**0**	b_2
a_{31}	a_{32}	**0**	0	**0**	**1**	b_3

2.2.2.3 Algoritmus řešení

Princip řešení simplexového algoritmu spočívá v opakování několika kroků. Pro lepší představu přikládám obrázek 2.1 Schéma Simplexového algoritmu.

Zmínění obrázek schématu popíši po jednotlivých krocích. Při popisu využiji příklad z knihy Ekonomicko-matematické metody[1] pro lepší znázornění obecného modelu lineárního programování, který obsahuje dvě rozhodovací proměnné a tři omezující podmínky.

$$a_{11}x_1 + a_{12}x_2 \leq b_1$$

$$a_{21}x_1 + a_{22}x_2 \leq b_2$$

$$a_{31}x_1 + a_{32}x_2 \leq b_3$$

$$z = c_1x_1 + c_2x_2 \rightarrow MAX$$

$$x_1, x_2 \geq 0 \; [1]$$

Podmínky Simplexového algoritmu celý algoritmus řešení vychází z předchozí Jordanovy eliminační metody. Navíc je odvození testu optimality i testu přípustnosti. To znamená, že celý Simplexový algoritmus musí mít nejen omezující podmínky, ale i kanonické tvary spolu s nezápornými hodnotami na pravých stran vektoru b.

„Následující schéma pro převod modelu do kanonického tvaru platí univerzálně:

- *podmínku* $a_{11}x_1 + a_{12}x_2 \leq b_1$ *převedeme na ...* $a_{11}x_1 + a_{12}x_2 + d_1 = b_1$*;*
- *podmínku* $a_{21}x_1 + a_{22}x_2 \geq b_2$ *převedeme na ...* $a_{21}x_1 + a_{22}x_2 - p_2 = b_2$*;*
- *podmínku* $a_{31}x_1 + a_{32}x_2 = b_3$ *převedeme na ...* $a_{31}x_1 + a_{32}x_2 + p_3 = b_3$ “ [1, strana 36]

Pro lepší přehlednost zmíněný obecný model zapíši do tabulky 2.1 Jednotková submatice v matici soustavy omezujících podmínek, kvůli znázornění požadovaného jednotkového vektoru.

Tabulka 2.2: Obecná simplexová tabulka[1]

		c_1	c_2	0	0	M	M		
$\boldsymbol{c_B}$	$\boldsymbol{x_B}$	x_1	x_2	d_1	d_2	p_2	p_3	$\boldsymbol{b}$	Ω
0	d_1	a_{11}	a_{12}	1	0	0	0	b_1	Ω_1
M	p_2	a_{21}	a_{22}	0	-1	1	0	b_2	Ω_2
M	p_3	a_{31}	a_{32}	0	0	0	1	b_3	Ω_3
$z_j - c_j$		$z_1 - c_1$	$z_2 - c_2$	$z_3 - c_3$	$z_4 - c_4$	$z_5 - c_5$	$z_6 - c_6$	z	

Výchozí bázické řešení z toho plyne, že tvary soustav určují bazické řešení. Zmíněné řešení obsahuje n proměnných a m omezujících podmínek. Bazické proměnné se považují v kanonické tvaru soustavy příslušné jednotkové vektory. Ostatní proměnné považujeme za nebazické. Hodnoty bázických proměnných jsou rovny hodnotám pravých stran.[2] Ostatní proměnné označujeme jako nebázické a jejich hodnotu je pokládána rovná nule.[1] Z toho plyne, že výsledek vyplývá ze závěrečné fáze bázického řešení.

„Pokud má úloha lineárního programování přípustné řešení, má také přípustné bázické řešení.

Pokud existuje optimální řešení úlohy LP, potom také nutně existuje optimální bázické řešení úlohy LP.“[1, stana 37]

Celý příklad budu řešit pomocí převedení do kanonického tvaru. Poté mám všechny potřebné podklady pro sestavení 2.2 Obecné simplexové tabulky.

V Prvním řádku tabulky se nacházejí ceny všech proměnných, ve druhém názvy proměnných a dalších pomocných sloupců. Obsah sloupců zleva doprava tvoří: vektor cen proměnných, které jsou v bázi, názvy těchto bázických proměnných, vektory technicko-ekonomických koeficientů u proměnných v omezujících podmínkách (matice soustay) a vektor pravých stran. Poslední řádek je vyhrazen testu optimality, poslední sloupec testu přípustnosti.[1]

„Nebázické proměnné pokládáme rovny nule, bázické proměnné proto nabývají hodnot vektoru pravých stran. Pro naši obecnou tabulku by platilo:

x_1 je nebázická, její hodnota je rovna 0;

x_2 je nebázická, její hodnota je rovna 0;

d_1 je bázická, její hodnota je rovna b_1;

d_2 je nebázická, její hodnota je rovna 0;

p_2 je bázická, její hodnota je rovna b_2;

p_3 je bázická, její hodnota je rovna b_3;

Řešení můžeme alternativně zapsat pomocí vektoru bázického řešení

$$x_B = (0; 0; b_1; 0; b_2; b_3)$$

nebo jako vektor obecného řešení:

$$x_0 = \begin{pmatrix} x_1 \\ x_2 \\ b_1 - a_{11} - a_{12} \\ d_2 \\ b_2 - a_{21} - a_{22} + d_2 \\ b_3 - a_{31} - a_{32} \end{pmatrix}$$ "[1, strana 39]

Jak lze vypozorovat z řešení, že je rozsah reálných procesů (aktiv) nulový, neboli $x_1 = 0$ a $x_2 = 0$. Tím nedošlo k čerpání prvního zdroje, protože rezerva d_1 se rovná původní kapacitě zdroje b_1. Dále nebyl překročen požadavek ve druhé podmínce $d_2 = 0$. Hodnoty pomocných proměnných p_2 a p_3 udávají, že do naplnění minimálního požadavku ve druhé omezující podmínce chybí b_2 jednotek a pro dosažení přesného požadavku ve třetí omezující podmínce chybí b_3 jednotek.[1]

Vzniklé řešení je nepřípustné, neboť nesplňuje všechny omezující podmínky, kvůli bázi příkladu zůstala pomocná proměnná.

„Konečně je možné vypočítat hodnotu účelové funkce dosazením hodnot všech proměnných do jejího předpisu jako skalární součin vektoru cen bázických proměnných a vektoru pravých stran, tedy

$$z = 0.b_1 + M.b_2 + M.b_3$$"[1, strana 39]

Test optimality je zapotřebí zjistit jestli bázické řešení je možné zlepšit či nikoliv. Tuto akci provádíme na začátku každé iterace. Jedná se výpočet kriteriálních hodnot $(z_s - c_s)_{min}$ nebo $(z_s - c_s)_{max}$ za pomocí Jordanovy eliminační metody. Jestliže se hodnoty $= 0$, pak celý algoritmus končí a nalezený modul řešení je optimální. Pokud je pro $(z_s - c_s)_{min} \geq 0$, resp. $(z_s - c_s)_{max} \leq 0$, nalezená hodnota x_s se přidá do báze.

Nyní budu pokračovat v ilustračním příkladu. Začnu se zařazením promněnné x_1 na míro promněnné d_1. Celá eliminace proběhne přes prvek a_{11}. Podle Jordanovy eliminační metody nejprve vydělíme celý první řádek hodnotou a_{11} a poté od druhého a třetího řádku odečteme a_{21}-násobek resp. a_{31}-násobek nového prvního řádku.[1] Tím dostaneme následující hodnoty v tabulce 2.3 Simplexová tabulka po změně báze.

„Pro původní tabulku dostaneme:

$$z^{(1)} = c_3 b_1 + c_5 b_2 + c_6 b_3,$$

pro druhou

$$z^{(2)} = c_1 \frac{b_1}{a_{11}} + c_5 \left(b_2 - a_{21} \frac{b_1}{a_{11}} \right) + c_6 \left(b_3 - a_{31} \frac{b_1}{a_{11}} \right),$$

po úpravě

$$z^{(2)} = c_1 \frac{b_1}{a_{11}} + c_5 b_2 - c_5 b_2 - a_{21} \frac{b_1}{a_{11}} + c_6 b_3 - c_6 a_{31} \frac{b_1}{a_{11}}$$

Tabulka 2.3: Simplexová tabulka po změně báze[1]

		c_1	c_2	c_3	c_4	c_5	c_6	
$\boldsymbol{c_B}$	$\boldsymbol{x_B}$	x_1	x_2	d_1	d_2	p_2	p_3	$\boldsymbol{b}$
c_1	d_1	1	$\frac{a_{12}}{a_{11}}$	$\frac{1}{a_{11}}$	0	0	0	$\frac{b_1}{a_{11}}$
c_5	p_2	0	$a_{22} - a_{21}\frac{a_{12}}{a_{11}}$	$-a_{21}\frac{1}{a_{11}}$	-1	1	0	$b_2 - a_{21}\frac{b_1}{a_{11}}$
c_6	p_3	0	$a_{32} - \frac{a_{12}}{a_{11}a_{31}}$	$-a_{31}\frac{1}{a_{11}}$	0	0	1	$b_3 - a_{31}\frac{b_1}{a_{11}}$

Rozdíl obou hodnot účelových funkcí vypočteme jako

$$\Delta z = z^{(2)} - z^{(1)} = c_1\frac{b_1}{a_{11}} + c_5 b_2 - c_5 b_2 - a_{21}\frac{b_1}{a_{11}} + c_6 b_3 - c_6 a_{31}\frac{b_1}{a_{11}} - c_3 b_1 - c_5 b_2 - c_6 b_3,$$

z toho

$$\Delta z = \frac{c_1 b_1}{a_{11}} - \frac{c_3 a_{11} b_1}{a_{11}} - \frac{c_5 a_{21} b_1}{a_{11}} - \frac{c_6 a_{31} b_1}{a_{11}},$$

a dále

$$\Delta z = -\frac{b_1}{a_{11}}\left(-c_1 + c_3 a_{11} + c_5 a_{21} + c_6 a_{31}\right),$$

Z rozdílu účelových funkcí vidíme, že se na změně hodnoty účelové funkce podílejí pouze tři faktory:

- *nová hodnota proměnné, která vstupuje do báze, zde* $x_1 = \frac{b_1}{a_{11}}$*;*
- *cena proměnné, která vstupuje do báze, zde cenový koeficient* c_1*;*
- *skalární součin vektoru cen proměnných, které tvořily bázi výchozích řešení, a koeficientů v matici soustavy pod testovanou proměnnou, zde výraz* $c_3 a_{11} + c_5 a_{21} + c_6 a_{31}$*“*[1, strana 40]

Nyní lze zjišťovat test optimality, u kterého provedu výhodnost či nevýhodnost nebázických proměnných.

$$„z_j - c_j = \boldsymbol{c_B}.\boldsymbol{a_j} - c_j,$$

kde
$\boldsymbol{c_B}$ *je vektor cen bázických proměnných v tabulce s testovaným řešením;*
$\boldsymbol{a_J}$ *je vektor matice soustavy pod testovanou proměnnou a*
c_j *je cena testované proměnné.“*[1, strana 41]

Výsledné řešení je optimální, pokud při maximalizaci již nebude existovat lepší nebázické řešení s vyšší hodnotou účelové funkce. Reps. pro minimalizaci nebude existovat nebázická proměnná s nižší hodnotou účelové funkce. Pokud stále řešení není optimální, i tak je zapotřebí vybrat novou nebázickou proměnnou. Je potřebné vybrat nejlepší zlepšující proměnnou. Neboli absolutní hodnotu testu optimality.

Test přípustnosti dále se provede test přípustnosti, při kterém se vypočítají hodnoty $\Omega_{min}^{(s)}$. Je zapotřebí zlepšit hodnotu účelové funkce. Mohou nastav dvě varianty existence hodnoty. Při neexistenci, nelze najít konečné optimální řešení a zároveň výpočet končí. Naopak při existenci hodnoty $\Omega_{min}^{(s)}$ se proměnná x_r vyřadí z báze. Nově nalezené hodnoty musí splňovat všechny omezující podmínky.

Smysl testu přípustnosti představím při pokračování v příkladu v jednom kroku Jordanovy eliminační metody. Z tabulky 2.3 Simplexová tabulka po změně báze zjistím, že v novém vektoru pravých stran se vyskytují následující hodnoty:

$$\text{v prvním řádku } \frac{b_1}{a_{11}} \text{ pro proměněnou } x_1$$

$$\text{pro ostatní řádky } b_i - a_{i1}\frac{b_1}{a_{11}}, \text{ kde i je číslo řádku.}$$

„Využijeme skutečnost, že všechny prvky ve vektoru pravých stran musí být nezáporná čísla. Proto v prvním případě stačí zajistit, aby hodnota prvku a_{11} byla kladná. Nulou dělit nelze, a pokud bychom kladnou hodnotu b_1 vydělili záporným koeficientem a_{11}, ihned bychom v novém vektoru pravých stran dostali záporné číslo.

Pro všechny ostatní řádky musí platit, že

$$b_i - a_{i1}\frac{b_1}{a_{11}} \geq 0,$$

po úpravě

$$\frac{b_i}{a_{i1}} \geq \frac{b_1}{a_{11}}.$$

Pro všechny řádky porovnáme poměry příslušné hodnoty z vektoru pravých stran b_i a hodnoty v matici soustav v klíčovém sloupci a_{i1}.“
[1, strana 42]

Ze zmíněné podmínky vyplývá, že poměr musí mít lepší nebo stejnou hodnotu, než v poměru proměnné x_1. Stále musí platit první podmínka kladné hodnoty jmenovatele.

Přechod na nové bázické řešení s lepší hodnotou účelové funkce pro přechod na nové bazické řešení, je zapotřebí provést jeden krok Jordanovy eliminační metody. Vybraný prvek α_{rs}, který nalezneme ve sloupci proměnné x_s a v řádku x_r z báze vyřadím. Celý postup se opakuje od části testu optimality.

Obecné schéma pro ruční výpočet Simplexové metody existuje schéma Simplexové tabulky. Jedná se o rozšířenou matici soustav lineárních rovnic zapsanou do tabulky. Navíc je v tabulce sloupce a řádky pro hodnoty obou kritérií. Jak, lze vidět na schématu tabulky 2.4 Schéma obecné Simplexové tabulky.

Výsledek řešení simplexového modelu v následujících podkapitolách představím možná řešení. K těmto řešením lze dospět v průběhu výpočtu. Jedná se o čtyři typy výsledku.

Tabulka 2.4: Schéma obecné simplexové tabulky

		x_1	x_2	x_3	...	x_n		
		c_1	c_2	c_3	...	c_n		$\Omega_{min}^{(k)}$
X_{B1}	C_{B1}							β_{B1}/α_{B1}
X_{B2}	C_{B2}							β_{B2}/α_{B2}
X_{B3}	C_{B3}			A			b	β_{B3}/α_{B3}
$\vdots$	$\vdots$							$\vdots$
X_{Bm}	C_{Bm}							β_{Bm}/α_{Bm}
$(Z_r - c_r)_{min}$		$z_1 - c_1$	$z_2 - c_2$	$z_3 - c_3$	...	$z_n - c_n$	$z(x)$	

Model nemá žádné přípustné řešení řešení, které není přípustné ve smyslu nesplnění omezujících podmínek, poznáme snadno; báze úlohy v takovém případě obsahuje alespoň jednu pomocnou proměnnou s nenulovou hodnotou. Pokud nastane tato situace, ale test optimality signalizuje, že dané řešení nelze zlepšit, potom model nemá žádné přípustné řešení.[1]

Hodnota účelové funkce může neomezeně růst v tomto případě přípustné řešení existuje, a dokonce existuje i přípustné bázické řešení. Problém je v tom, že se kvůli množině přípustných neohraničené ve směru růstu účelové funkce časem dostaneme do situace, kdy nejsme schopni nalézt bázické řešení s lepší, ale omezenou hodnotou účelové funkce.[1]

Model má právě jedno optimální řešení v tomto případě stačí splnění dvou podmínek: řešení musí splňovat všechny omezující podmínky (tj. v bázi úlohy nesmí být pomocná proměnná) a test optimality musí signalizovat, že neexistuje způsob, jak hodnotu účelové funkce vylepšit. Aby optimální řešení bylo právě jedno, všechny nebázické proměnné musí hodnotu účelové funkce ztrácet svůj význam.[1]

Model má nekonečno mnoho optimálních řešení v terminologii simplexového algoritmu tato situace nastane v případě, že test optimality je sice splněn, ale existuje nebázická proměnná s hodnotou testu optimality rovné nule. Zařazení takové proměnné do řešení změní sice změní bázi úlohy, ale hodnota účelové funkce se nezmění. Model má nekonečně mnoho optimálních řešení: dvě bázická a nekonečně mnoho nebázických.[1]

2.2.2.4 Úprava lineárního modelu pro simplexový algoritmus

Simplexová metoda řeší pouze úlohu lineárního programování ve vhodném tvaru. Omezující podmínky rovnice tvaru, v kanonickém tvaru a hodnoty vektorů pravých stran, které nejsou záporné. Pokud nesplňují zmíněné podmínky je potřeba každý lineární model převést do vhodného tvaru.

Nezápornost hodnot pravých stran poměrně snadno na zajištění, stačí zápornou rovnici či nerovnici vynásobit -1. Pouze v případě nerovnice, je nutné zajistit, aby byla pořád zobrazena se správným smyslem.

Převedení libovolného typu omezujících podmínek do kanonického tvaru celá myšlenka je založena na umělém rozšíření počtu proměnných v modelu. Je nutné vytvořit novými proměnnými požadovaný tvar soustavy omezujících podmínek. Celý postup závisí na výchozím tvaru omezujících podmínek.

Omezující podmínka lineárního modelu je nerovnice typu „$\leq$“ lze hovořit o kapacitní podmínce. Hlavním cílem zmíněné podmínky je vyrovnání rozdílu pravé a levé strany rovnice. Jedná se o doplnění nezáporné proměnné, čímž vznikne kanonický tvar. Nově vzniklá proměnná se nazývá doplňovací proměnná typu rezerva, vyjadřující rezervu dané kapacity.

Když budou podmínky ve zmíněných požadovaných tvarech dostaneme:

$$Ax + Ed = b, d \geq 0,$$

$$\textit{kde } \; d = (d_1, \ldots, d_m)^T$$

pak vektor bázického řešení je

$$x^{(0)} = (0, \ldots, 0 | b_1, \ldots, b_m)^T \text{ .[2]}$$

Omezující podmínka lineárního modelu je nerovnice typu „$\geq$“ jedná se o požadavkové podmínce, která je splněna po doplnění nezáporné proměnné. Ta vyjadřuje překročení požadavku a může být převedena do rovnicového tvaru. Nově vzniklá proměnná se nazývá doplňková proměnná typu překročení. Pro správné použití zmíněné proměnné se musí odečíst od levé strany podmínky. Pak vznikne výraz, který není v kanonickém tvaru, protože vektory nejsou jednotkové, ale obsahují hodnotu -1 místo 1

Když budou podmínky ve zmíněných požadovaných tvarech dostaneme:

$$Ax + Ed = b, d \geq 0,$$

$$\textit{kde } \; d = (d_1, \ldots, d_m)^T \text{ [2]}$$

Uvedený výraz neobsahuje jednotkovou matici jako u minulého případu. Můžeme vynásobit každou rovnici soustavy -1 a tím dostaneme kanonický tvar soustavy, až na záporné hodnoty pravých stran vektorů bazického řešení

Omezující podmínka lineárního modelu je v rovnici, ale není v kanonickém tvaru hovoříme o podmínkách, které byly již v rovnicovém tvaru od formulace modelu, nebo do tohoto tvaru byli převedené, či doplněné doplňkových proměnných. Na nově vzniklou podmínku je nutná rozšířit o pomocné proměněné, tak aby pomohla vytvořit kanonický tvar.

„Pokud byly všechny omezující podmínky rovnice, vektory koeficientů pomocných proměnných vytvoří kanonickou bázi a dostaneme

$$Ax + Ep = b, p \geq 0,$$

kde $p = (p_1, \ldots, p_m)^T$ *jsou pomocné proměnné*

Považujeme-li v této soustavě pomocné proměnné p *za bázické, dostaneme vektor bázického řešení*

$$x^{(0)} = (0, \ldots, 0 \mid b_1, \ldots, b_m)^T .$$

jehož složky jsou nezáporná čísla, což je v souladu s předpoklady, že $b \geq 0, x \geq 0, p \geq 0$.

Pokud byly všechny omezující podmínky požadavkového typu, vektory koeficientů pomocných proměnných tvoří kanonickou bázi a dostaneme

$$Ax - Ed + Ep = b, d \geq 0, p \geq 0,$$

kde $p = (p_1, \ldots, p_m)^T$ *jsou pomocné proměnné*

Považujeme-li v této soustavě pomocné proměnné p *za bázické, dostaneme vektor bázického řešení*

$$x^{(0)} = (0, \ldots, 0 \mid 0, \ldots, 0 \mid b_1, \ldots, b_m)^T .$$

jehož složky jsou nezáporná čísla, což je v souladu s předpoklady, že $b \geq 0, d \geq 0, x \geq 0, p \geq 0$. "[2, strana 74]

Zmíněné doplňkové proměněné při využití souvisí s ekonomickým výrazem překročení požadavku, nebo rezervy kapacity. Poté dostanu ekvivalenci lineárních soustav, včetně omezujících podmínek, po rozšíření o doplňkové proměnné je ekvivalentní mezi soustavy.

Z ekonomické interpretace vyplývá též nulové ocenění doplňkových proměnných v účelové funkci.[2] V účelové funkci koeficient c_j vyjadřuje přínosovou proměnou. Nulová cena jsou nevyužité výrobní zdroje a překročení jejich požadavků. Cena nemá žádný užitek, proto všem doplňkovým proměněným připisuji nulovou hodnotu.

Soustava omezujících podmínek rozšířená o alespoň jednu pomocnou proměnnou není ekvivalentní s původní soustavou ani se soustavou rozšířenou pouze o doplňkové proměnné. Pokud pomocné proměnné jsou rovny nule, pak je možné získat řešení původního modelu. Vynecháním pravé strany pomocným proměnným. Proto je nutné během výpočtu všechny pomocné proměnné vyřadit z řešení tak, aby v optimálním řešení byly všechny pomocné proměnné nebázické, tj. aby jejich hodnoty byly nulové.
[2]

Z výpočtu vyřadím pomocné proměnné a tím docílím nevýhodnost pomocných proměnných. Pomocné proměnné vždy ohodnocujeme nevýhodou, tzv. prohibitivní sazbou účelové funkci.[1] Při maximalizaci bude prohibitivní cena velké záporné číslo (např. -1000, -10000 atd.), při minimalizaci velké kladné (např.

1000, 10000 atd.). Většinou musí tyto sazby být řádově vyšší než koeficienty ostatních proměnných modelu[2] Může nastat, že zůstane v optimálním řešením nějaká pomocná proměnná, pak to není přípustné řešení pro daný problém. V uvedeném případě dostanu prázdnou množinu řešení. Prohibitní sazbu značíme symbolem M.[1]

2.2.3 Analýza optimálního řešení a post optimalizační analýza

„Optimální řešení udává optimální stav systému v určitém okamžiku a při splněné určitého souboru předpokladů popsaných soustavou omezujících podmínek a účelovou funkcí.“[1, strana 60]

Optimum, které získám většinou nestačí k rozhodnutí optimality. Nalezení matematického optima souvisí s pevně danými kvantitativními údaji. Zmíněné údaje se mohou v průběhu výpočtu měnit proto rozhodnutí závisí na tzv. kvalitativních vlivech. Bohužel nejdou popsat lineárním optimalizačním modelem, ale mohou velkým způsobem měnit podmínky rozhodnutí. Většinou ke změnám dojde v důsledku vzájemných vlivů modelovaného systému nebo zapříčiněným náhodných činitelů. Cílem analýzy optimálního řešení je získat další informace o rozhodování v daném systému.

Z výše uvedených důvodů je potřeba provést důkladný analytický rozbor informací z výpočtu lineárního modelu. Mezi nejdůležitější činnosti patří úplný popis získaného optimálního řešení a rozbor všech údajů. Získané informace slouží ke změnám rozhodnutí při změně vstupních podmínek celého problému. Zmíněný postup se nazývá postoptimalizační analýza, která umožňuje přesně určit rozsah změn a postupů v rámci přípustného řešení.

2.2.3.1 Zobrazení údajů výsledné simplexové tabulky

„Vyhodnocení výsledné Simplexové tabulky rozumíme v podstatě přečtení výsledku, jeho ekonomickou interpretaci a vypsání všech informací, které poskytuje výsledná simplexová tabulka bez zásahů do výchozích údajů.“
[2, strana 78]

Podle konkrétní situace se provádí rozsah vyhodnocení Simplexové tabulky. Tím získám jednotlivý přehled vyhodnocených bodů. Záleží na konkrétním případě, zda budu mít všechny uvedené provedené operace. Nejdůležitější je mít původní simplexový model pro vyhodnocení získaných údajů.

Optimální řešení Řešení, které je optimální x_{opt}, získám z výsledné simplexové tabulky. Dále dostanu optimální hodnotu účelové funkce z_{opt}

$$x_{opt} = (X_B, X_N)_{opt} = (\beta_1, \ldots, \beta_m, 0, \ldots, 0)^T \text{ [2]}$$

Optimální hodnota účelové funkce je:

$$z_{opt} = c^T x_{opt}. \text{ [2]}$$

Matice transformace B^{-1} Ve výsledné simplexové tabulce naleznu matici transformace B^{-1}. Nachází se na pozicích jednotkových vektorů v původní tabulce.

„Její význam spočívá v tom, že lze s její pomocí transformovat úlohy z výchozí do výsledné tabulky v jediném kroku.“[1, stránka 45]

Duální hodnoty slouží k výpočtu kritéria optimality $(z_j - c_j)$[1] pro výsledné řešení modelu. Cíl duální hodnot je udávat, jak a o kolik se zhorší hodnota účelové funkce. V případě nevyužití řeším zlepšení využití zmíněného činitele.

2.2.3.2 Vliv změn hodnot nebazických proměnných na optimální řešení

Alternativní řešení je takové řešení, které podle hodnoty účelové funkce je optimální. Tím, že zařadím do řešení nebázické proměnné x_j s nulovou duální hodnotou $(z_j - c_j) = 0$[2]. Dochází k alternativnímu řešení.

Suboptimální řešení je takové řešení, ve kterém alespoň u jedné z nebázických proměnných x_j přiřadíme nenulovou hodnotu. Změnou založenou na změně hodnot účelové funkce pomocí duálních hodnot je se jedná o lepší řešení. Vzniklé suboptimální řešení bude blíž k optimálnímu v závislosti na blízkosti k nule u nebázických proměnných $(z_j - c_j)$[2].

Nelze vybrat libovolnou hodnotu a tu dosadit do řešení, protože by mi mohlo vzniknout nepřípustné řešení. V případě, že budu řešit pouze jedinou nebázickou proměnnou x_k, pak řešení zůstane přípustné. Pokud požadovaná hodnota x_k splňuje podmínku intervalu přípustných hodnot:

$$0 \leq x_k \leq \Omega_{min}^{(k)} \text{ neboli } x_k \in \left\langle 0, \Omega_{min}^{(k)} \right\rangle \text{ [2]}$$

Pokud dosadím za x_k dolní mez, tj. nulovou hodnotu, tak se nezmění řešení. Při dosažení horní mezi, tj. $x_k = \Omega_{min}^{(k)}$, dostanu nové bázické řešení, kde se účelová funkce zhorší o $-\Omega_{min}^{(k)}(z_k - c_k)$.

Když budu rozšiřovat řešení o více nebázických proměnných $x_k, k \in K$, pak musí splňovat zachování hodnot přípustnosti řešení podmínky:

$$\beta_i - \sum\nolimits_{k \in K} \alpha_{ik} x_k \geq 0, i = 1, \ldots, m.\text{[2]}$$

2.2.4 Analýza citlivosti

Cílem analýzy citlivosti je určení rozsahu změn výchozích hodnot údajů lineární optimalizační úlohy. Během této činnosti nedochází ke změnám optimální báze. Platí pravidlo čím je větší rozsah změn, tím je stabilnější optimální řešení a naopak.

Pokaždé se vychází z nalezeného optimálního řešení, kde se sledují změny při změně hodnot vstupních parametrů. Změny se předpokládají ve tvaru:

$$\tilde{\alpha} = \alpha + \lambda\text{[2]},$$

kde $\lambda \in \left\langle \underline{\lambda}, \overline{\lambda} \right\rangle$ je interval možných změn původní hodnoty parametru α

„Na tomto místě se budeme zabývat nejčastějšími postoptimalizačními úvahami, které se v rámci řešení modelů LP provádí:

- *zařazením nebázického procesu do řešení a tedy tvorbou nebázického řešení;*
- *analýzou stability řešení vzhledem ke změnám hodnot pravých stran;*
- *analýzou citlivosti optimálního řešení vzhledem ke změnám cen.“*[1, strana 60]

2.2.4.1 Zařazení nebázické proměnné do řešení

V bázickém řešením je hodnota nebázické proměnné rovna nule. Zařadím nejlepší maximální úroveň pro nebázické proměnné do řešení, tak aby hodnoty vektorů pravých stran zůstaly nezáporné. Jedná se o hledání intervalu přípustných hodnot nebázické proměnné.

„Dolní mez tohoto intervalu je kvůli podmínkách nezápornosti rovna nula. Pro zjištění horní meze stačí využít standardní nástroj Simplexového algoritmu: testu přístupnosti. Při zařazování výhodné proměnné do báze test určuje maximální přípustnou hodnotu této proměnné. Zařazujeme do řešení proměnnou, která z hlediska účelové funkce nebude výhodná, nicméně na jeví maximální přípustnou hodnotu to nemá žádný vliv.

Pokud bychom tedy chtěli do řešení zařadit nebázickou proměnnou x_k, její interval přípustných hodnot bude:

$$x_k \in \langle 0, min_{i=1\ldots m}\Omega_i \rangle$$

Zařazení nebázické proměnné do řešení bude mít vliv na hodnoty bázických proměnných a na hodnotu účelové funkce.“[1, strana 61]

2.2.4.2 Analýza citlivosti vzhledem ke změně jedné složky vektoru pravých stran b

Nejprve se začnu zabývat rozsahem změn optimálního řešení, při změnách výchozích požadavků a kapacit, neboli změnami pravých stran omezujících podmínek.

„Změnu hodnot pravých stran vyjádříme:

$$b(\lambda) = b + \lambda,$$

takže každou hodnotu pravé strany vyjádříme jako:

$$b_i(\lambda) = b_i + \lambda_i, i = 1, \ldots, m.$$

Vektor parametrů $\lambda = (\lambda_1, \ldots, \lambda_m)^T$ se složky λ_i vyjadřující změnu příslušné složky b_i.

Nechť B^{-1} je matice transformace příslušné k matici optimální báze B. Řešení úlohy je pak dáno vztahem:

$$x_B(\lambda) = B^{-1}(b + \lambda) = B^{-1}.b + B^{-1}.\lambda = \beta + B^{-1}.\lambda$$

neboli:

$$\beta_i(\lambda) = \beta_i + \sum_{k \in B} \alpha_{ik}.\lambda_k, i = 1, \ldots, m.$$

Kde α_{ik} jsou složky k-tého sloupce matice B^{-1} .

V případě že se mění právě jediná složka vektoru b, nechť složka b_k bude mít výchozí soustava omezujících podmínek lineární úlohy tvar:

$$Ax = b + \lambda_k.e_k \ ,$$

a řešením této soustavy bude:

$$x_B(\lambda) = B^{-1}(b + \lambda_k.e_k) = B^{-1}.b + B^{-1}.\lambda_k.e_k = \beta + B^{-1}.\lambda_k.\alpha_k$$

kde e_k je jednotkový vektor,

α_k je k-tý sloupec matice B_{-1} .

Aby bylo řešení v bázi B stále přípustné, musí být splněna podmínka:

$$\beta_i + \lambda_k \alpha_{ik} \geq 0, i = 1, \ldots, m \ .$$

*Řešení této soustavy nerovností je **interval stability** řešení, resp. **interval přípustných hodnot** parametrů λ_k:*

$$\lambda_k \in \left\langle \underline{\lambda}, \overline{\lambda} \right\rangle = \left\langle max_{\alpha_{ik}>0} \left(\frac{-\beta_i}{\alpha_{ik}} \right), min_{\alpha_{ik}<0} \left(\frac{-\beta_i}{\alpha_{ik}} \right) \right\rangle \ .$$

Pokud neexistuje $\alpha_{ik} > 0$, resp. $\alpha_{ik} < 0$, klademe $\underline{\lambda} = -\infty$, resp. $\overline{\lambda} = \infty$.“[2, strana 81]

Změna hodnot pravé strany nezmění vliv na charakter optimálního řešení. Mění se pouze optimální rozsahy procesů tak i konkrétní hodnota kritéria.

2.2.4.3 Analýza citlivosti vzhledem ke změně jedné složky vektoru cen c

Nyní se začnu zabývat rozsahem změn optimálního řešení, při změnách cenových koeficientů jednotlivých proměnných.

Změnu hodnot pravých stran vyjádříme:

$$c(\lambda) = c + \lambda,$$

takže každou cenu můžeme vyjádřit jako:

$$c_j(\lambda) = c_j + \lambda_j, j = 1, \ldots, m.$$

Vektor parametrů $\lambda = (\lambda_1, \ldots, \lambda_m)^T$ *se složky* λ_i *vyjadřující změnu příslušné složky* c_j.

Abychom mohli zjisti, kdy nedojde ke změně optimálního řešení, spočítáme hodnoty v kriteriálním řádku v závislosti na vektoru parametru λ*:*

$$z_j(\lambda) - c_j(\lambda) = c_B^T(\lambda)\alpha_j - c_j(\lambda) =$$

$$c_B^T\alpha_j - c_j + \lambda_B\alpha_j - \lambda_j = z_j - c_j + \lambda_B\alpha_j - \lambda_j \ .$$

V případě ***maximalizační úlohy*** *musí i tyto parametrické hodnoty zůstat nezáporné:*

$$z_j - c_j + \lambda_B\alpha_j - \lambda_j \geq 0.$$

Změnou složek cenového vektoru, upravíme ceny nebázické proměnné x_j*, poté vektor* λ *bude obsahovat jako jedinou nenulovou hodnotu* λ_j*, která nebude v části* λ_B*. Z toho plyne, že celá změna se projeví jenom v kriteriální hodnotě proměnné* x_j*.*

$$z_j - c_j - \lambda_j \geq 0 \ ,$$

pro stálost optimálního řešení musí platit:

$$\lambda_j \leq z_j - c_j \ .$$

Dostaneme tak ***interval pro změnu cen u nebázické proměnné*** $\boldsymbol{x_j}$ *ve tvaru:*

$$\lambda_j \in \left\langle \underline{\lambda}, \overline{\lambda} \right\rangle \in \langle -\infty, z_j - c_j \rangle \ .$$

V případě změny pouze u ceny bázické proměnné x_j*, dostaneme nenulový parametr* λ_j*, který bude v části změnového vektoru* λ_B*. U nebázických proměnných v kriteriálním řádku získám podmínku:*

$$z_j - c_j + \lambda_i\alpha_{ij} \geq 0 \text{ pro každé } j = 1, \ldots, n,$$

ze kterých dostaneme ***interval stability pro změnu cen u bázické proměnné****:*

$$\lambda_j \in \left\langle \underline{\lambda}, \overline{\lambda} \right\rangle = \left\langle max_{\alpha_{ij}>0}\left(\frac{-z_j + c_j}{\alpha_{ij}}\right), min_{\alpha_{ij}<0}\left(\frac{-z_j + c_j}{\alpha_{ij}}\right) \right\rangle$$

Pokud neexistuje $\alpha_{ij} > 0$*, resp.* $\alpha_{ij} < 0$*, klademe* $\underline{\lambda} = -\infty$*, resp.* $\overline{\lambda} = \infty$.“[2, strana 83]

Změna ceny nemá vliv na charakter optimálního řešení. Nemění se optimální rozsahy procesů, pouze se mění konkrétní hodnota kritéria.

„V případě ***minimalizační úlohy*** *musí i tyto parametrické hodnoty zůstat nekladné, takže interval stability pro změnu ceny nebázické proměnné bude parametrické hodnoty:*

$$\lambda_j \in \left\langle \underline{\lambda}, \overline{\lambda} \right\rangle \in \langle z_j - c_j, \infty \rangle \ ,$$

a pro změnu ceny bázické proměnné dostaneme:

$$\lambda_j \in \left\langle \underline{\lambda}, \overline{\lambda} \right\rangle = \left\langle max_{\alpha_{ij}<0}\left(\frac{-z_j + c_j}{\alpha_{ij}}\right), min_{\alpha_{ij}<0}\left(\frac{-z_j + c_j}{\alpha_{ij}}\right) \right\rangle \ .“[2, strana 83]$$

2.3 Elektronické vzdělávání

Anglický ekvivalent e-learning se běžně i u nás používá. Jedná se o formu vzdělávání využívající moderní informační a komunikační technologie.
Osoby zapojené do elektronického kurzu částečně nebo zcela přes vzdálený přístup. E-learning má různé podoby v závislosti na stanovených cílech výuky či na potřebách studenta. Může sloužit k předávání výukových materiálů a obsahů komunikace mezi účastníky kurzu i k řízení vzdálené výuky.

V koronavirové době během roku 2020 se elektronické vzdělávání masivně rozšířilo, od zakládaný až po vysoké školy. Určitá forma distanční výuky již probíhala před koronavirovou uzávěrkou kontaktní výuky. Lze mluvit o tom, že během zmíněné doby mnoho škol prošlo rozsáhlou digitalizací a rozvoji online výuky. Podle mého názoru distanční výuka nahradila kontaktní výuku, pro ty studenty, kteří měli zájem o učení. Ale stejně je lepší přímí kontakt s vyučujícím, něž celý den koukat do obrazovky.

Do elektronického vzdělávaní patří systém Moodle. Zmíněnému systému se budu věnovat v následující kapitole. Neboť ve své práci na něm budu pracovat a připravovat zlepšení testování.

Kromě Moodlu existují i další systémy na vzdálenou výuku.

2.4 Moodle

Obrázek 2.2: Logo Moodle
Zdroj: https://moodle.org/

Jedná se o otevřený systém na podporu výuky za pomocí online nástrojů. Lze využít k různým způsobům, např. bezkontaktní výuku. Jak lze vypozorovat i z obrázku 2.2 Loga Moodle. Systém nabízí možnost založení samostatného kurzu na konkrétní potřebnou činnost. Zmíněný kurz nabízí různé moduly, jedná se zejména o přednášky, testy, úkoly, dotazníky, docházku, známky atd.

Hlavně se zaměřím na tvorbu testování, včetně jeho vytváření. V této kapitole budu především čerpat z dokumentace k Moodlu. Existují dvě oficiální dokumentace, jedna v čestině[5] a druhá rozsáhlá v angličtině[6].

2.4.1 o systému

Nejprve začnu definicemi zmíněného systému Moodle. *„Název Moodle [čteme múdl] se už i u nás v Česku natolik vžil, že jeho jméno běžně skloňujeme a jeho aktivním programátorům říkáme moodlisté, případně moodláci.“*[7, strana 14]

Poté představím zajímavosti. Následně se budu věnovat otevřenosti, včetně používané licence.

2.4.1.1 Definice

Existují různé definice pro systém Moodle, kterým se budu zaobírat.

„Název systému Moodle je akronymem pro modulární objektově orientované dynamické vzdělávací prostředí (Modular Object-Oriented Dynamic Learning Environment).“[8, strana 11]

V překladu modulární objektově orientované dynamické výukové prostředí. Z tohoto důvodu, lze tvrdit o volně dostupném systému na úpravu.

Podle dokumentace oficiální definice: *„Moodle je výuková platforma navržená tak, aby poskytovala pedagogům, administrátorům a studentům jediný robustní, bezpečný a integrovaný systém pro vytváření personalizovaných výukových prostředí.“*[6, Hlavní strana ▶ o Moodlu]

Podle mého názoru je podstatné, že celý Moodle slouží především ke zlepšení výuky. Za pomocí různých komponent, kterým se budu dále věnovat.

2.4.1.2 Dostupnost

Celý vytvořený systém je dostupný pro jednoduché používání vzdělávání a výuky, neboli e-learning.

„Moodle účelně využívá současné webové stránky a serverové technologie, přičemž od uživatele nevyžaduje zvláštní počítačovou zručnost.“
[8, strana 12]

Pro plnohodnotné využití stačí uživateli pouze osobní počítač, tablet, či chytrý telefon s webovým prohlížečem. Dále je zapotřebí připojení k internetu pro přístup ke vzdálenému serveru.

2.4.1.3 Licence

„Moodle je poskytován zdarma jako Open Source, který spadá pod obecnou veřejnou licenci GNU.“[7, strana 12] Jedná se o dílo chráněné autorskými právy se svobodnými úpravami. Ve své bakalářské práci[9] jsem se zabýval otevřenou licencí MIT, která vychází z GPL. Rozdíl mezi zmíněnými otevřenými licencemi je: *„MIT lze použít jako proprietární software nebo GPL licencovaný software při zachování textu podmínek.“*[9, strana 61]

Pokud využiji GPL kód do svého projektu, tak se automaticky stává GPL. Jinak řečeno, vytvořím otevřený zdrojový kód (Open Source), to znamená, že lze nejen číst kód, ale i upravovat či přidávat vlastní kód.

2.4.1.4 Požadavky systému na tvorbu

„Moodle je primárně vyvíjen v Linuxu pomocí Apache, MySQL a PHP (někdy známé jako platforma LAMP). Obvykle je to také způsob, jakým se Moodle spouští, i když existují i jiné možnosti, pokud jsou splněny softwarové požadavky v závislosti na dané verzi.“
[6, Hlavní stránka ▶ Instalace ▶ Instalace Moodle]

Potřebujeme minimální hardwarové požadavky na počítač. Minimálně 160 MB volného místa na pevném disku a 256 MB volné operační paměti.[8] Dále se nesmí zapomenout na uložené dokumenty, studijní materiály, soubory, atd. pro jednotlivé kurzy, které zabývají také volné místo. Podle
dokumentace[6] min. 5 GB volného místa.

U uložení instalace na serveru je zapotřebí větší operační i dostupná paměť, protože je potřebné počítat s více uživateli najednou. *„U velkých instalací Moodlu (řádově stovky současně pracujících uživatelů) doporučujeme, aby byl Moodle provozován na dvou serverech - na jednom bude spuštěn webový server spolu s daty a na druhém pouze databázový server.“*[8, strana 252]

Zmíněná konfigurace je hlavně finančně náročná.

Linux je operační systém, licence Linuxu je GPL, jako Moodle[10]

Apache je webový server. Webový server je program, služebník, který našemu prohlížeči posílá webové stránky a nám se pak v prohlížeči

zobrazují.[10] Projekt Apache je snahou vyvinout a udržovat Open-source HTTP server pro moderní operační systémy včetně UNIX a Windows. Cílem tohoto projektu je poskytnout bezpečný, efektivní a rozšiřitelný server, který poskytuje služby HTTP synchronizované se současnými standardy HTTP.[11]

MySQL je databázový server. Právě sem se ukládaná data vložená uživateli. [10] MySQL je celosvětově nejoblíbenější open source databáze. MySQL vám může nákladově efektivním způsobem pomoci poskytovat vysoce výkonné a škálovatelné databázové aplikace.[12]

PHP je populární univerzální skriptovací jazyk, který je zvláště vhodný pro vývoj webových aplikací. Rychlé, flexibilní a pragmatické, PHP pohání vše od Vašeho blogu až po nejoblíbenější webové stránky na světě.[13] Všechny změny se projeví na serveru, mezi jeho výhody patří změna kódu v textovém editoru, která se okamžitě projeví na webové stránce.

2.4.1.5 Jazyk

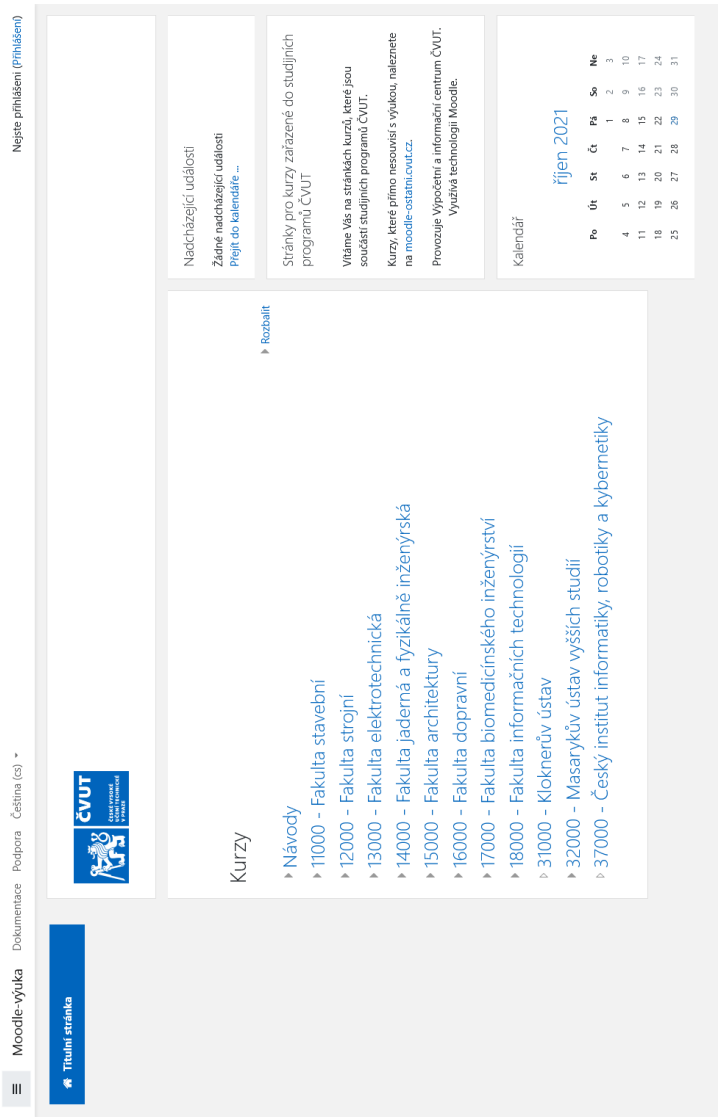
Podle dokumentace[6] je více než 100 jazykových balíčků. Podle mého názoru je nejdůležitější čeština. Mezi světovým jazykům jako je angličtina, němčina a další jazyky.

2.4.1.6 Záloha

Zálohování patří k nejdůležitější věci. Moodle nabízí sám od sebe automatickou zálohou jednotlivých kurzů. Samozřejmostí je ruční záloha kurzů i dat. Je možné mluvit o tom, že zálohovat lze v podstatě všechno.

2.4.2 Navigace v systém

Při spuštění výukových stránek, např. Moodle ČVUT dostupné na adrese: `https://moodle-vyuka.cvut.cz/` Pro lepší orientaci přikládám obrázek 2.3 Ukázka úvodní obrazovky Moodle. Navštěvuji stránku jako host, neboli nepřihlášený uživatel. V této roli vidím pouze základní obecné informace. Jako jsou kategorie kurzů podle jednotlivých fakult a po rozkliknutí se zobrazí seznam fakult s konkretními kurzy.



Obrázek 2.3: Ukázka úvodní obrazovky Moodle
Zdroj: snímek obrazovky Bc. Denise Drdy z Moodle ČVUT

Po přihlášení do systému se nacházím na úvodní stránce tvz. **Nástěnka**. Nyní je potřebné si položit otázku: *„Kudy vede nejkratší cesta do jednotlivých kurzů?“*[7]. Proto na své úvodní obrazovce jsou v základní instalaci pro ČVUT dva bloky **Nedávno navštívené kurzy**, **Studijní plány** a **Přehled kurzů**. Nutno

podotknout, že je možné zmíněné základní nastavení různě upravovat a přizpůsobovat svým požadavkům. Po přístupu do systému si mohu vybrat konkrétní kurz, který chci navštívit.

2.4.3 Role

Role určují uživateli rozsah pravomocí v systému Moodle. Každá role má jiné možnosti především v zobrazování úprav, editaci, atd. Např. role učitele umožňuje vkládat zadání úkolů do kurzu a odevzdaná řešení hodnotit. Pouze v roli student může odevzdat své řešení. *„V některých případech lze mechanismus rolí využít k tomu, abyste změnili výchozí chování Moodlu. Tak můžete například ve svém kurzu udělit několika vybraným studentům oprávnění hodnotit úkoly ostatním.“*[5, Kategorie: Role]

Host může si prohlížet kurzy, ale nemůže se jich účastnit.[6]

Student nutná registrace má přístup k prohlížení obsahu zapsaných kurzů. [14]

Učitel bez práv úprav (Asistent) může v kurzech hodnotit, ale nemůže je upravovat. [6]

Učitel s právy úprav (Učitel) může spravovat a přidávat obsah do kurzů. [6]

Tvůrce kurzu (Garant) obvykle shodný s garantem předmětu, garant může měnit nastavení a uspořádání svých kurzů a přidělovat role vyučujícím. [15]

Správce (GAELP) garant elektronické podpory výuky za katedru, GAELP může spravovat kurzy a vytvářet nové kurzy za katedru.[15]

Správce stránek (Administrátor) má volnost pro práci všeho

2.4.4 Režim úprav

Jedná se o speciální režim, ve kterém je možné editovat kurz i jeho obsah. Lze rozšiřovat uživatelské rozhraní, měnit obsah rozložení a dalšího. Pro zmíněnou úpravu je nutné zapnout tento režim pomocí tlačítka „Zapnout režim úprav“. Pro lepší přestavu přikládám obrázek 2.4 Tlačítko na zapnutí režimu úprav.

Obrázek 2.4: Tlačítko na zapnutí režimu úprav
Zdroj: snímek obrazovky Bc. Denise Drdy z DEMO Moodlu

2.4.5 Formáty textů

Podle české[5] a anglické[6] dokumentace Moodlu je možné psát texty v různých formátech. Lze psát i ve výchozím nastavení a nic neměnit. Zmíněné řešení je vhodné pro začátečníky. Další složitější formáty textů představím níže.

2.4.5.1 Autoformát

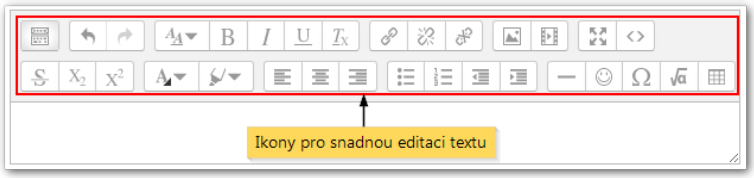
Lze využít běžné textové pole pro editaci a psaní textů. Nejedná se vestavěný HTML editor, který bude v následujícím sekci. Např. lze přirovnat, k psaní e-mailu.

„Po uložení provede Moodle s textem automaticky několik operací:

- *Internetové adresy ve tvaru https://docs.moodle.org/cs/ atp. jsou převedeny na odkazy*
- *Zalomení řádků je ignorováno, další odstavec začneme vložením prázdného řádku*
- *Smajlíky jako :-) se převedou na grafický ekvivalent*
- *Můžeme vložit HMTL kód, který bude zachován“*
  [5, Kategorie: Učitel ▶ Formátování textu]

2.4.5.2 HTML editor

Pro použití zmíněného editoru využitím přijatelného uživatelského prostředí, umožnující provádět základní editace. HTML editor nabízí různé funkce, které jsou velmi blízké textovému editoru Microsoft Word. Jak lze vidět na obrázku 2.5 Ukázka HTML editoru.



Obrázek 2.5: Ukázka HTML editoru
Zdroj: `http://moodledocs.phil.muni.cz/editace-textu/html-editor`

Dále je možné z výběru mezi ikony pro snadnou editaci různě upravovat daný text. Nejen editovat odsazení textu, ale i vkládat různé symboly, či smajlíky. Lze vidět na obrázku 2.6 Ruční vložení smajlíků.

2.4.5.3 Čistě textový formát

Je vhodný pro vložení programového kódu či HTML textu bez následujících úprav. Znamená to, že text bude vyobrazen stejně jako je napsán. Pouze mezery a úpravy v zalomení řádků jsou ignorovány. Ostatní části textu jsou nedotčeny.

Obrázek 2.6: Ruční vložení smajlíků
Zdroj: snímek obrazovky Bc. Denise Drdy z DEMO Moodlu

2.4.5.4 Formát Markdown

Umožnouje vytvářet formátované XHTML stránky. Psaní ve zmíněném formátu má své náležitosti a syntaxi. *„Je velmi dobrý pro psaní stránek s čistým textem obsahujícím několik nadpisů a odrážkových seznamů bez mnoha odkazů nebo obrázků. Je to nejlepší volba, pokud je pro vás důležitá dostupnost generované stránky.“*
[5, Kategorie: Učitel ▶ Formátování textu]

2.4.5.5 Vkládání matematiky

U textového editoru je potřebné zapnout režim **TeX zápis**, resp. **Algebraický zápis** umožňuje vkládat matematické vzorce a výrazy.

Např. pro vložení výrazu $x_1^2 + x_2^2$ v kódu TeX se matematický výraz zapisuje pomocí symbolu $$ (dvou dolarů), bude následující kód:

```
$$x_1^2+x_2^2$$
```

Syntaxe pro algebraický zápis má velmi podobnou syntaxi, ale místo dolarů jsou použity dva zavináče @@. Pro stejný výraz $x_1^2 + x_2^2$, bude kód následující:

```
@@x_1^2+x_2^2@@
```

2.4.6 Kurz

Jedná se o nejdůležitější zastřešující část v systému Moodle. Kurz sdružuje celou cílenou aktivitu na internetu, lze si tento pojem představit jako název předmětu. Umožňuje učitelovi přidávat různé výukové aktivity, vytvářet a zveřejňovat materiály.

V kurzu jsou různé sekce, mezi které patří nastavení kurzů, ale i správu uživatelů. Tímto pojmem je definován zápis studentů do kurzů a rozdělení do skupin. Je možné i celý kurzu archivovat a poté jej následně obnovit.

2.4.7 Moduly

„Obsah kurzu se sestavuje z tzv. modulů. Každý modul má své specifické vlastnosti a nastavení a širokou škálu možností využití.“[5, Kategorie: Moduly činností ▶ $Moduly$]

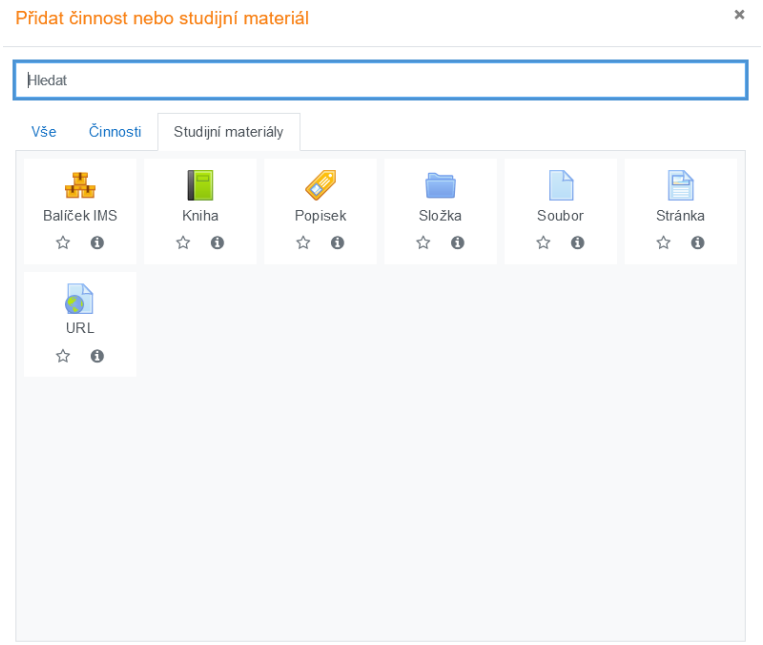
Moduly jsou základní stavební prvek v celém systému. Mají širokou možnost využití. Mohou sloužit ke studium, např. k vkládání studijních textů nebo jako podpora studia (přes možné diskuze nad odbornými články, po odevzdávání úkolů a absolvování testů).

2.4.7.1 Studijní materiály

Studijní materiály neboli angl. *Resource* jsou nenormálním typem modulů. Celý systém umožňuje podporu různých typů a formátů studijních materiálů, které lze vložit do kurzu. Zobrazit studijní materiál může kdokoliv, ale pouze v roli učitele je možné vkládat a editovat soubory.

Pro přesun či přidání nového studijního materiálu je potřebné zapnout režim úprav. *„Přidáním i jen jednoho studijního materiálu se současně v postranním bloku Činnosti zobrazí odkaz „Studijní materiály“, který vede na seznam všech studijních materiálů vložených do daného kurzu.“*
[5, Kategorie: Moduly činností ▶ Studijní materiál]

Poté je možné vložit nové dokumenty na požadované místo, za pomocí rozbalovacích nabídek „Přidat studijní materiál. . .“ Jak je vidět na obrázku: 2.7 Rozbalovací nabídka pro přidání studijního materiálu.



Obrázek 2.7: Rozbalovací nabídka pro přidání studijního materiálu
Zdroj: snímek obrazovky Bc. Denise Drdy z DEMO Moodlu

Balíček IMS umožňuje do kurzu vložit obsah ve formátu dle specifikace IMS Content Packaging. Balíček IMS může být použit pro prezentaci multimediálního obsahu a animací.[15]

Modul Kniha tento typ studijního materiálu umožňuje přímo v Moodlu snadno vytvořit kompletní webovou stránku pomocí vestavěného HTML editoru i bez znalosti jazyka HTML.[5]

Popisek umožňuje do osnovy kurzu vložit text, obrázky a multimédia mezi odkazy na další činnosti[15]

Složka umožňuje učiteli zobrazit několik souvisejících souborů v jedné složce a pomáhá tak snížit potřebu rolování na hlavní stránce kurzu. Větší množství najednou lze nahrát pomocí ZIP archívu.[15]

Soubor umožňuje učiteli poskytnout soubor jako studijní materiál.[15]

Stránka dovoluje vkládat obsah ve formě čistého neformátovaného textu. To, jak bude takový text zobrazen studentům, je ovlivněno zvoleným formátem textu (Moodle auto-formát, HTML, čistý text nebo Markdown
syntaxe).[5]

Modul URL dovoluje vložit odkaz na soubor nahraný do souborového manažeru v kurzu (např. PDF, DOC, XLS apod.) nebo hypertextový odkaz na webovou stránku či jiný zdroj, který je možno adresovat pomocí URL.[5]

2.4.7.2 Aktivity/Standardní moduly činnosti

V této podkapitole stručně představím jednotlivé moduly. U každého modulu je možné různé nastavení na základě různorodých specifických vlastností. K hlavním vlastnostem Moodlu patří jeho využití při práci studentů.

„Studijní materiály jsou chápány jako zázemí či podklad. Samo učení však podle tvůrců Moodlu spočívá v aktivním zapojení se do širokého spektra vzdělávacích aktivit, v jejichž průběhu si studenti konstruují nové poznání.“[5, Kategorie: Učitel ▶ Moduly činností]

Moodle lze využít jako obyčejný redakční systém pro prezentaci studijních materiálů na internetových stránkách.

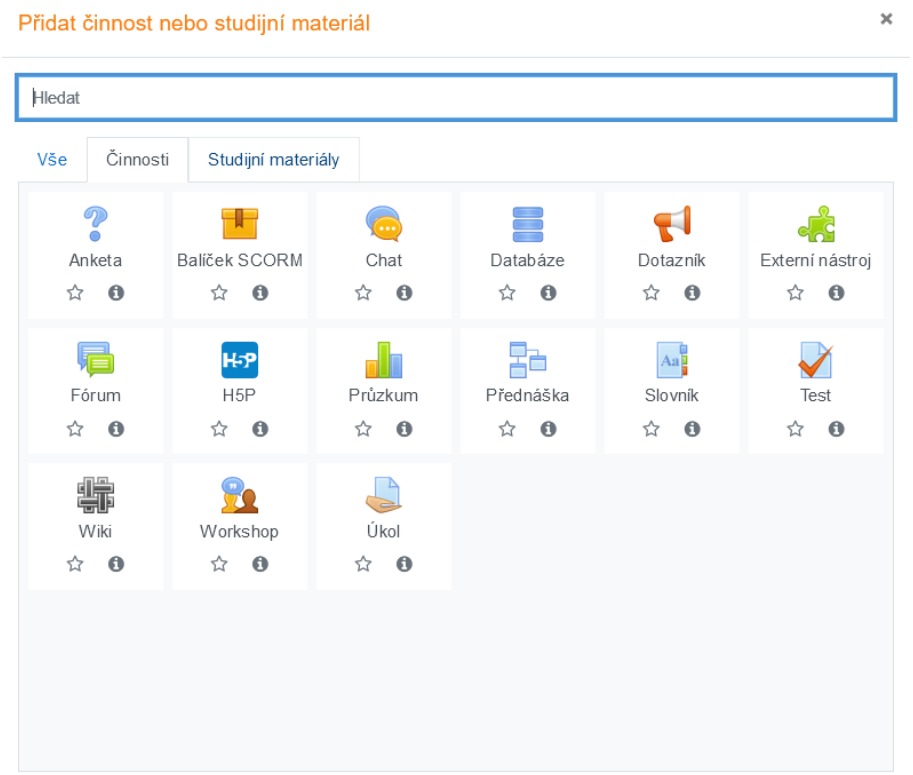
V anglické dokumentaci[6] se uvádějí aktivity, jedná se pouze o odlišný překlad vůči české dokumentaci[5] se nazývají standardní moduly činností činnosti.

„Aktivita je obvykle něco, co student bude dělat a které interaguje s ostatními studenty nebo učitelem. V terminologii Moodle aktivita, jako jsou fóra nebo kvízy, správně znamená něco, k čemu mohou studenti přímo přispět, a je často v kontrastu se zdrojem , jako je soubor nebo stránka, který jim učitel předloží. Termín aktivita je však někdy pro pohodlí používán také pro označení aktivit a zdrojů jako skupiny.“
[6, Hlavní stránka ▶ Správa kurzu Moodle ▶ Aktivity]

Hlavním cíl je potenciál k podpoře učení se studenta pomocí různých typů činností. Pouze v roli učitele mohu vkládat do kurzu jednotlivé moduly. Pro úpravu či přidání nového modulu je zapotřebí zapnout režim úprav. Poté je možné přidat několik činností, jak je vidět na obrázku 2.8 Rozbalovací nabídka pro přidaní činnosti. Následně stručně představím zmíněné činnosti.

Anketa umožňuje učiteli položit otázku s několika odpověď'mi, z nichž si studenti mohou vybrat. To umožňuje uskutečnit rychlé hlasování, kterým lze například podnítit studenty k přemýšlení o určitém tématu, nechat je rozhodnout o dalším postupu v kurzu nebo mezi nimi provést průzkum mínění.[5]

Balíček SCORM balíček SCORM umožňuje do kurzu vložit obsah ve formátu dle specifikací SCORM a AICC. Jedná se o soubory, které jsou zabaleny podle standardu pro výukové objekty.[16]



Obrázek 2.8: Rozbalovací nabídka pro přidání činnosti
Zdroj: snímek obrazovky Bc. Denise Drdy z DEMO Moodlu

Chat umožňuje účastníkům kurzu diskutovat na webu synchronně v reálném čase.[15]

Databáze modul Databáze umožňuje vytvářet, prohlížet a prohledávat kolekci položek (tj. Záznamů). Strukturu příspěvků definuje učitel jako řadu polí. Typy polí zahrnují zaškrtávací políčko, přepínače, rozevírací nabídku, textovou oblast, adresu URL, obrázek a nahraný soubor.[6]

Dotazník modul Dotazník umožňuje realizovat dotazníkové šetření. Slouží pro získání zpětné vazby od účastníků kurzu. Při návrhu dotazníku lze použít řadu typů položek včetně výběru z daných hodnot nebo volné tvořené odpovědi.[16]

Externí nástroj modul Externí nástroj umožňuje studentům pracovat s prostředky a činnostmi na jiných webových stránkách. Externí nástroj může například poskytovat přístup k novému typu činnosti nebo k výukovým materiálům od jiného vydavatele.[16]

Fórum zde nejčastěji probíhá diskuse mezi účastníky kurzu. Fóra mohou být uspořádána několika různými způsoby a mohou zahrnovat hodnocení příspěvků ostatními účastníky kurzu či učitelem.[5]

H5P je zkratka pro balíček HTML5 - interaktivní obsah, jako jsou prezentace, videa a další multimédia, dotazy, kvízy, hry a další. Činnost H5P umožňuje nahrát a přidat H5P do kurzu.[16]

Průzkum poskytuje několik vestavěných dotazníkových nástrojů, které se osvědčily při hodnocení a stimulaci výuky v on-line prostředí. Učitelé je mohou používat ke sběru dat, z nichž se mohou dozvědět více o svých studentech a mít zpětnou vazbu při své výuce.[5]

Přednáška ta umožňuje vytvářet adaptivní a interaktivní výkladový materiál. Učitel může využít Přednášku pro vytvoření posloupnosti stránek nebo vzdělávacích aktivit, které nabízejí studentovi celou řadu cest a možností procházení.[15]

Slovník umožňuje účastníkům kurzu vytvářet a průběžně spravovat seznam definic. Hesla lze vyhledávat a zobrazovat v mnoha různých formátech.[5]

Test zde vkládáme testy skládající se z výběrových úloh, dichotomických úloh, srovnávacích úloh, úloh s krátkou tvořenou odpovědí a dalších[15]

Úkol učiteli může zadávat úlohy, jejichž splnění vyžaduje, aby student vytvořil digitální obsah (v libovolném formátu) a uložil ho na server. Typickými úkoly jsou eseje, projekty, referáty atd. Modul obsahuje také nástroje pro hodnocení.[5]

Wiki umožňuje účastníkům kurzu společně vytvářet a editovat webové stránky, zakládat je, rozšiřovat a měnit jejich obsah.[15]

Workshop umožňuje sběr a vzájemné hodnocení prací studentů.[15]

2.5 Test

V předchozí kapitole jsem stručně představil zmíněný modul. Jelikož se ve své diplomové práci budu věnovat vylepšení testování v Moodlu, tak je potřebné důkladně probrat testovací modul neboli testovací činnost. Lze také hovořit o kvízových úlohách (podle dokumentace) jedná se o stejný význam v závislosti na konkrétním překladu.

„V jazyce Moodle se modul Kvíz může nazývat testovací modul. Učitelé v Moodlu mohou označovat kvíz jako druh hodnocení studentů. Tuto definici lze sdílet slovem „test“.“[6, Hlavní stránka ▶ Test]

Existuje mnoho základních testových úloh, kterým se budu věnovat v následující podčástí.

„Kvíz je velmi silná aktivita, která může splnit mnoho výukových potřeb, od jednoduchých znalostních testů s více možnostmi až po složité úkoly sebehodnocení s podrobnou zpětnou vazbou.“
[6, Hlavní stránka ▶ Kvíz]

Vytvořené úlohy v rámci Moodlu jsou ukládané v Bance úloh, poté jsou vkládané do konkrétních testů.

Modul test umožňuje učiteli vytvářet a zadávat testy, skládající se z úloh různého typu: např. výběr z několika možností, pravda/nepravda, tvořená odpověď', krátká tvořená odpověď', přiřazování, numerická úloha, doplňovací úloha apod.[5] Každá vytvořená úloha je uložena v databázi. Ze které je možné ji znovu použít pro testování opakovaně. Učitel může nastavit počet povolených pokusů, úlohy zamíchat nebo náhodně vybírat z banky úloh. Může být nastaven časový limit pro testování.[16] Každý pokus je automaticky ohodnocen a učitel si může vybrat, zda k jednotlivým úlohám poskytne studentům komentář, nebo zobrazí správnou odpověď'. Modul obsahuje také nástroje pro známkování.[5]

„Testy mohou být použity

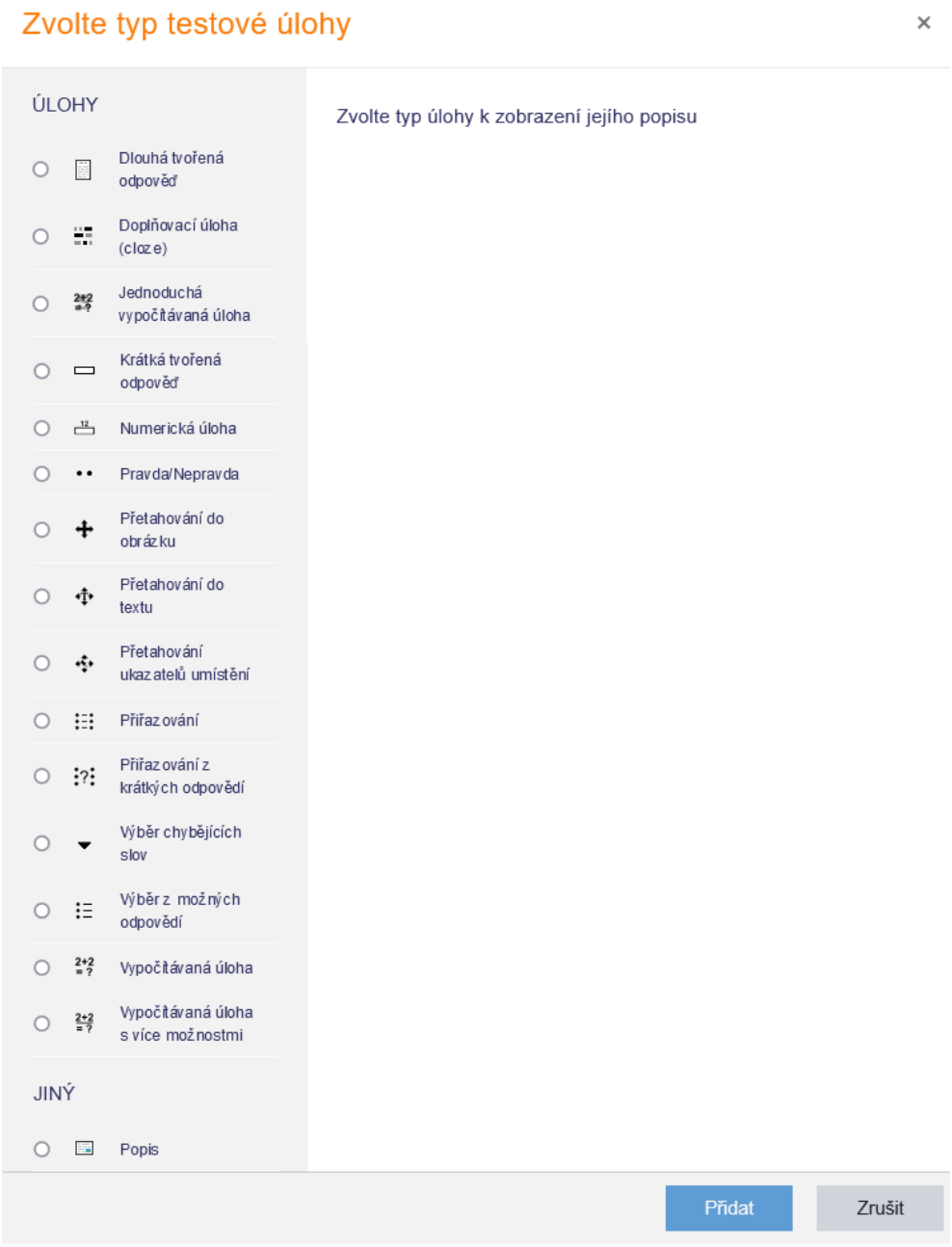
- *jako zkouška v kurzu*
- *jako mini test pro samostudium nebo na konci tématu*
- *v souborném testu s využitím úloh předchozích testů*
- *chcete-li zajistit okamžitý test výkonu*
- *pro sebehodnocení studenta“*
  [16, Přidání nové činnosti (Test) ▶ Pomoc s testem]

2.5.1 Typy testových úloh

V rámci modulů Test a Přednáška, lze přidávat různé typy testových úloh. Oba moduly jsou velmi podobné na testování, liší se pouze v počtu otázek a odlišného fungování mezi moduly. V modulu Testu je více možností testových sad.

K testové úloze je možné přidat i obrázek, či video. Dále můžeme vyplnit bodové ohodnocení.

Nyní uvedu standardní typy testovacích otázek se stručným popisem. Pro lepší přehlednost přikládám obrázek 2.9 Možnost zvolení typu testové úlohy.



Obrázek 2.9: Možnost zvolení typu testové úlohy
Zdroj: snímek obrazovky Bc. Denise Drdy z DEMO Moodlu

Dlouhá tvořená úloha je pro delší odpověď, ale přímo odpověď na složitější dotaz uvnitř testu, využijte tohoto typu. Odpověď studenta není automaticky ohodnocena, čeká se až na manuální ohodnocení učitelem.[5]

Doplňovací úloha umožňuje vytvořit část souvislého textu, který může obsahovat rozbalovací nabídky s volbou možností, políčka pro vložení tvořeného textu, výpočtu.[10]

Jednoduchá vypočítávaná úloha nabízejí způsob, jak vytvořit jednotlivé číselné otázky, jejichž odpověď je výsledkem číselného vzorce, který obsahuje proměnné číselné hodnoty pomocí zástupných znaků (tj. {x}, {y}), které jsou při kvízu nahrazeny náhodnými hodnotami.[6]

Krátká tvořená odpověď odpovědí na tento typ otázky je slovo, či slovní spojení, které studenti vepisují do připraveného pole.[5]

Numerická úloha se řadí k úlohám automaticky kontrolovaným systém. Umožní zadat otázku s číselnou odpovědí představující např. řešení daného příkladu.[10]

Pravda/Nepravda je jednoduchá úloha, kdy se ptáme na pravdivostní hodnotu[10]

Přetahování do obrázku obrázky nebo textové popisky jsou přetaženy do cílových zón na obrázek na pozadí.[16]

Přetahování do textu chybějící slova v textu se doplňují přetažením vybrané odpovědi do odpovídajícího prázdného pole.[16]

Přetahování ukazatelů umístění značky jsou přetaženy na obrázek na pozadí.[16]

Přiřazování k jednotlivým možnostem (části úlohy) přiřadili správnou odpověď.[8]

Přiřazování z krátkých odpovědí je jako přiřazovací úloha, ale je vytvořena náhodně z úloh s krátkou tvořenou odpovědí v dané kategorii.[16]

Výběr chybějících slov chybějící slova v textu se doplňují výběrem odpovědi z rozbalovací nabídky.[16]

Výběr z možných odpovědí umožňuje vytvářet automaticky opravované otázky s jednou nebo více možnými odpověďmi.[10]

Vypočítávaná úloha funguje stejně jako jednoduchá vypočítávaná úloha, ale v rámci tvorby úlohy umožňuje podrobnější nastavení, zejména v oblasti generování hodnot.[8]

Vypočítávaná úloha s více možnostmi se chová jako obyčejná úloha s více možnostmi, ale nabízené odpovědi se pro každého studenta vypočítávají jako výsledek daného vzorce s náhodně vybranými hodnotami z jisté množiny.[16]

Popis není opravdovou otázkou, slouží jako oddělovací, informační popisek, který může autor testu vkládat mezi jednotlivé otázky. Vyplněný Popis např. zobrazí jen vložený text (je možné vložit i grafiku nebo video), který slouží jako úvodní informace k podskupině následujících otázek.[5]

2.5.2 Banka úloh

Každý vytvořený test je složen z testových úloh nebo otázek. *„Vytvořené testové úlohy se ukládají na určité místo v e-learningovém kurzu, které nazýváme Bankou úloh.“*[10, strana 152]

Sekci Banku úloh lze nazývat databází úloh, navíc obsahuje kategorie úloh v rámci jednoho kurzu. Zmíněné úlohy slouží jako základní otázky pro test. Jednotlivé úlohy v Bance úloh je možné sdílet mezi uživateli, kteří mají možnost upravovat uvedenou sekci.

2.5.3 Kategorie úloh

„Úlohy jsou organizovány do kategorií. Každý kurz obsahuje zpočátku úvodní kategorii úloh nazvanou "Výchozí". Je vhodné vytvářet více kategorií, smysluplně je pojmenovávat, sdružovat podobné typy úloh a vytvářet hierarchickou strukturu.“[5, Kategorie: ► Úlohy ► Banka úloh]

Lze členit, organizovat i podle zaměření, složení i jiných charakteristik. Zmíněné úlohy je možné rozdělovat do kategorií a podkategorií. Celý systém zobrazování zvyšuje přehlednost a usnadňuje práci. Pro snaží pochopení, lze si představit úlohy jako soubory na počítači a kategorie jako složky, ve kterých jsou uloženy úlohy.

2.5.3.1 Ovládání a editace

Každou kategorii úlohy lze různě upravovat, či ji měnit mezi sebou. Lze i exportovat a importovat mezi kurzy. Samozřejmostí je i smazání z nabídky. Pro ovládání se používají šipky, např. pro přesun kategorie. Použití šipek pro přesun kategorií

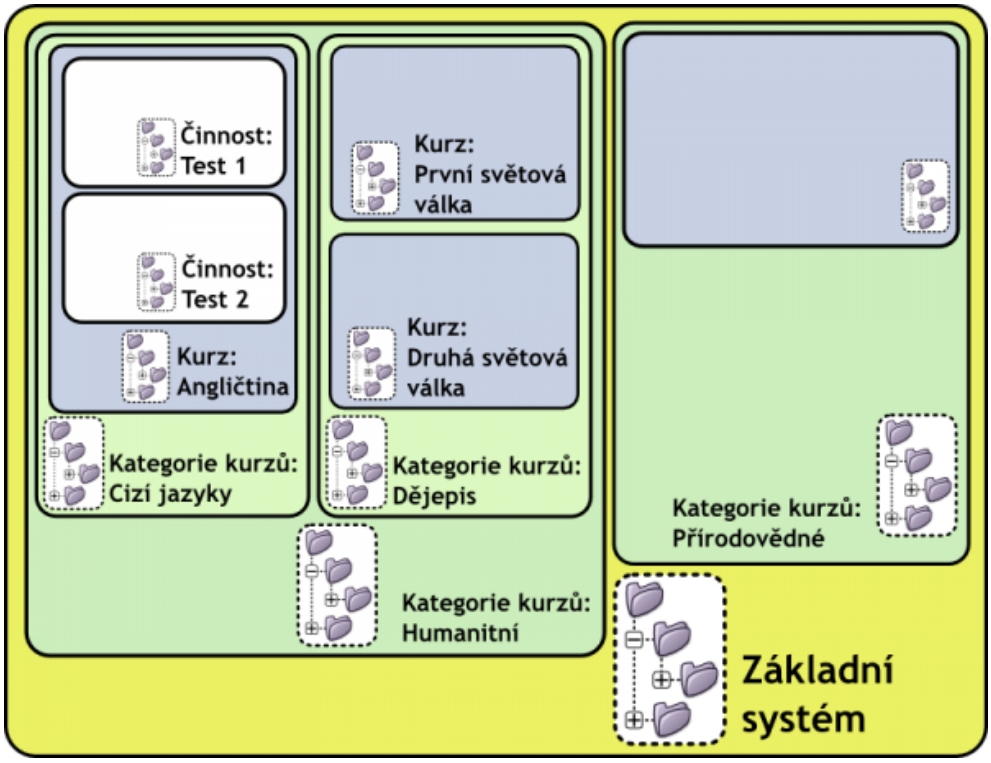
„Pro lepší přehlednost můžete přesouvat jednotlivé podkategorie pomocí šipek

- *Šipky nahoru/dolů mění pořadí zobrazení „sourozeneckých“ kategorií*
- *Šipka doleva přesouvá kategorii na vyšší úroveň. Pravá šipka pak přesouvá kategorii do jiné kategorie, tj. vytváří potomka*
  [5, Kategorie: ► Kategorie úlohy]

2.5.3.2 Kontexty úloh

„Můžeme sdílet otázky v různých kontextech. Kontext může být neznámé slovo. Představte si kontext jako oblast na vašem webu Moodle. Každý jiný kontext má samostatnou hierarchii kategorií otázek. Nyní si tedy můžete vybrat „kontext“, ve kterém budou vaše otázky sdíleny.“
[6, Hlavní stránka ▶ Správa kurzu Moodle ▶ Otázky ▶ Kontexty otázek]

Pro lepší představu přikládám obrázek 2.10 Příklad kontextů úloh: testové úlohy mohou být uloženy v několika různých oblastech vašeho serveru. Kontext je místo, ve kterém je strom kategorie úloh uložen. V základním nastavení má každý učitel právo vytvářet svou banku úloh pouze v kontextu jednoho kurzu (kontext kurzu je v diagramu vyznačen modrou barvou). Úlohy lze ale ukládat i do vyšších kontextů - např. do kontextu kategorie kurzů (odstíny zelené). Např. všechny úlohy uložené v kontextu „Dějepis“ mohou být použity v obou kurzech o světových válkách, ale neobjevují se ani v kurzech cizích jazyků, ani v kurzech přírodovědných.[5]



Obrázek 2.10: Příklad kontextů úloh: testové úlohy mohou být uloženy v několika různých oblastech vašeho serveru
Zdroj: `https://docs.moodle.org/archive/cs/Kontexty_%C3%BAloh`

2.5.4 Hodnocení

„Testování studentů je obvyklou formou jak zjistit úroveň znalostí studentů, ať už jako podklad pro další směřování výuky, nebo jako kontrolu toho, zda studenti dané problematice rozumí.“[10, strana 179]

Na prvním místě je hodnocení testu. Podle [10] se jedná, zda student uspěl či neuspěl, jakou známku nebo bodové hodnocení získal.

Podle mého názoru je spíše vhodnější slovní hodnocení. Hlavně je více osobní i motivační pro studenta. Jelikož se student dozví, kde konkretně se má zlepšit či v čem je dobrý. V e-learningu slovní hodnocení je spíše složitější, než automatické vyhodnocení. Výhodou ve zmíněném vyhodnocení je že stroj vyhodnotí úlohu sám a vyučující nemusí opravovat všechny testy.

2.5.5 Motivace

Obecně řečeno motivace se rozděluje na vnitřní a vnější. Motivace vnitřní patří k silnější motivaci, jedná se o poznávací potřebu, potřebu seberealizace, atd. Oproti tomu vnější motivace je způsobená vnějšími vlivy, jako je možnost odměny či hrozba trestu, atd.

„At' už chceme nebo ne, je naprosto vše přepočítáváno na peníze a od toho se často odvíjí i samotná motivace nejen studenta, ale také učitele nebo tvůrce e-learningu.“[10, strana 211]

Ale bohužel, zmíněná tvrzená nelze plnohodnotně využít pro studenty a motivací penězi. Proto je potřebné hledat cestu jak motivovat studenta za dobře odvedenou práci.

Tradiční rčení „Budeš-li se dobře učit, budeš se mít v životě dobře.“ je spojeno s příslibem skvělého budoucího života. Jedná se sice o motivaci se šťastným koncem, ale někdy v této motivaci během života vidíme protipříklad. Z tohoto důvodu by měla být celá motivace řízena netradičně, např. učením hrou.

2.6 Vypočítávaná úloha

„Vypočítávaná umožňuje jako odpovědi vyhodnocovat číselné údaje a nastavit pro ně toleranci. Na rozdíl od numerické úlohy v tomto případě definujete pro úlohu parametry, v rámci kterých systém náhodné hodnoty ze zadaného intervalu.“[8, strana 178]

Znamená to, že díky tomu budou různé hodnoty v zadání pro jednotlivé testy. Hodnoty pochází z předem nastaveného intervalu. V systému Moodlu probíhá výpočet i jeho následná kontrola.

2.6.1 Využití

Tato úloha může sloužit k výpočet různě obtížným problémů. Například se může jednat o „Výpočet plochy obdélníky“, zde můžu vytvořit otázku se dvěma zástupnými znaky (tj. a , b tvořené stranou a a b). Následně vložím do vstupního pole **Vzorec správné odpovědi**, kde * je znak násobení.

$$\textbf{Vzorec správné odpovědi} = \{a\} * \{b\}$$

2.6.2 Přidání a úprava

Nyní se zaměřím na vytváření vypočítávané úlohy. Zmíněný postup umožnuje i upravovat již hotovou úlohu. Níže představím nejdůležitější pokyny pro vyplnění krok za krokem. Pro lepší představu přikládám obrázek 2.11 Ukázka vypočítávané úlohy.

Obecná nastavení

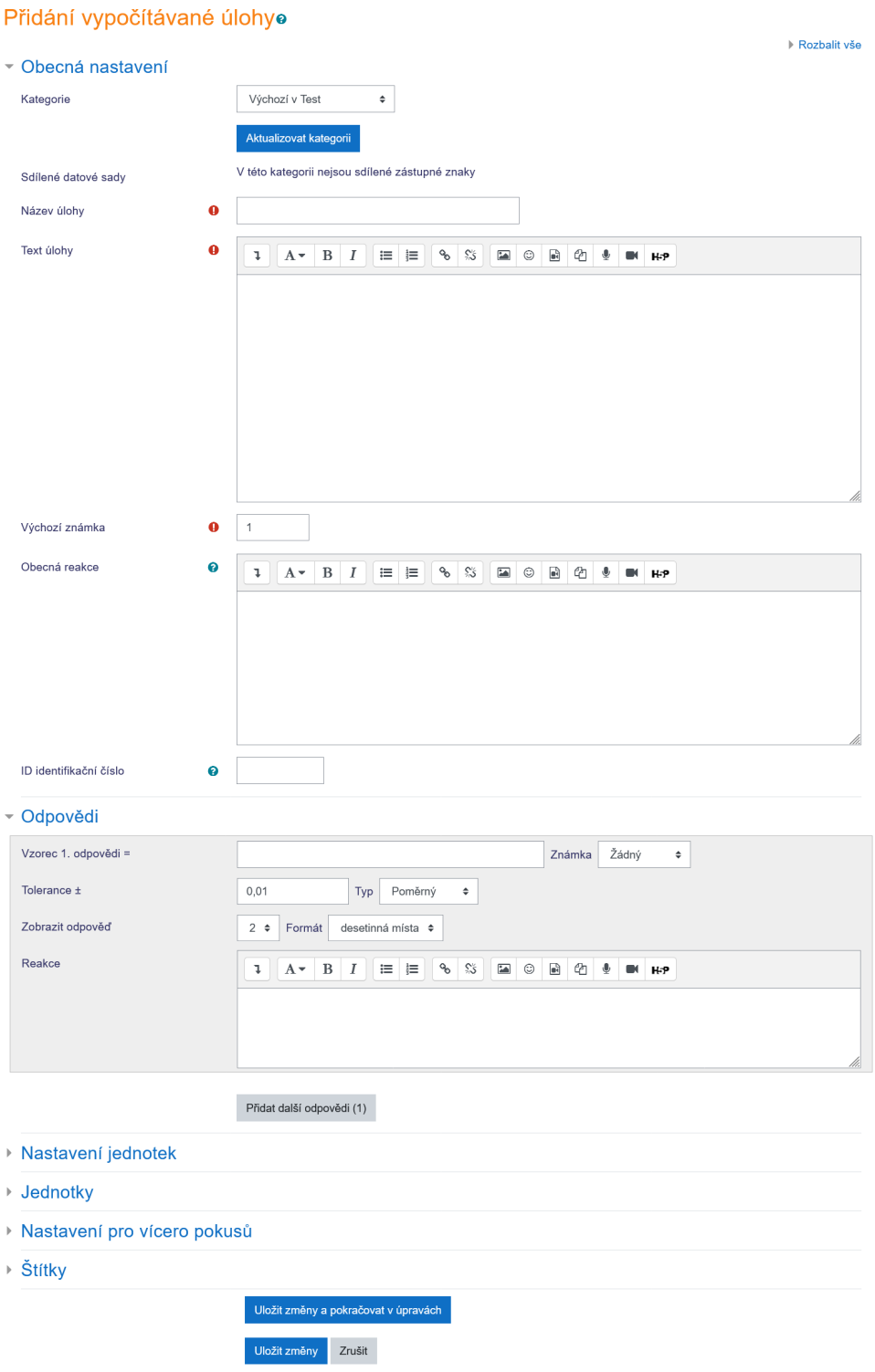
Kategorie Nejprve vyberu kategorii. *„Pokud změníte kategorii, budete muset kliknutím na tlačítko „Aktualizovat kategorii“ tento seznam obnovit. Možná ještě neexistují žádné sdílené zástupné znaky – pokud ne, můžete je vytvořit později, pokud chcete.“*
[6, Hlavní stránka ▶ Správa kurzu Moodle ▶ Úlohy ▶ Vypočítávaná úloha]

Název úlohy umožní identifikovat úlohu v Bance úloh.[6]

Text úlohy je text, který studentovi ukáže při zobrazení úkolu.[10]

Výchozí známka tj. maximální počet bodů pro tuto otázku.[6]

Obecná reakce je vzorec, který bude použit pro správný výpočet výsledku (ve většině případů bude totožný s textem úlohy) a tak stačí do políčka zadat vzorec správné úlohy.[10]



Obrázek 2.11: Ukázka vypočítávané úlohy
Zdroj: snímek obrazovky Bc. Denise Drdy z DEMO Moodlu

ID identifikační číslo je-li použito, číslo ID musí být v každé kategorii otázky jedinečné. Poskytuje jiný způsob, jak identifikovat otázku (může být zpravidla prázdná, někdy je užitečná).[16]

Odpovědi

Vzorec 1. odpovědi tento vzorec musí obsahovat alespoň zástupné znaky, které se objevují v textu otázky.[6]

Tolerance určuje toleranci chyby, kterou přijmáme v odpovědi. Nastavení tolerance a typu tolerance se kombinují a poskytují rozsah přijatelných skóre. [6]

Zobrazit odpověď' určuje formát a přesnost zobrazení.

Reakce je zpětná vazba, kterou student uvidí, pokud zadá tuto odpověď'.[6]

Přidat další odpověď' toto políčko mluví samo za sebe. Jednoduše se přidá další odpověď'.

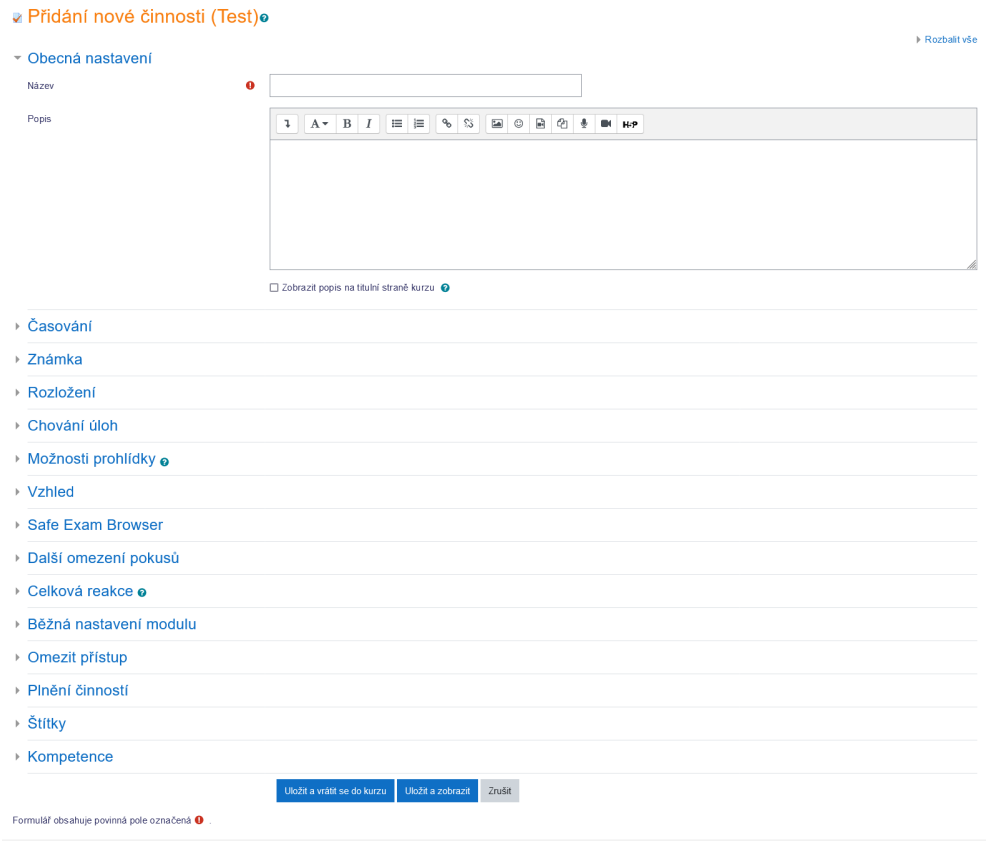
Další nastavení kde je možné nastavit jednotky zda jsou povinné nebo s volbou pro vyžadování při zadání výsledku. Zadat hodnotu penalizace v případě chybného výsledku. Dále je možné nastavit konkrétní jednotku. Různá nastavení pro vícero pokusů včetně penalizace za další pokus.

2.6.3 Datové sady

datová sada je množinou, odkud se určuje hodnota proměnné. Kde se nastavuje hlavně rozsah proměnné i počet desetinných míst. V textu a ve vzorcích je znázorňován pomocí proměnných, kde se proměnná doplní číslem. Datové sady je možné upravovat nejen při zadávání úlohy, ale pouze při změně údajů.

2.7 Vytváření testu v Moodlu

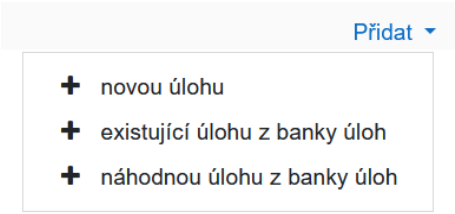
V této podkapitole se zaměřím na vytváření testů až po přidání testových úloh, včetně stručného popisu činnosti testu. Nejprve je potřebné zapnout režim úprav. Dále vložím novou aktivitu, konkrétně test, jak je vidět na obrázku 2.12 Přidání nové činnosti (Test). Kde lze nastavit od základních informací, jako je název testu, časový limit, zpřístupnění a uzavření testu, datum otevření až po hodnocení, revizi, atd.



Obrázek 2.12: Přidání nové činnosti (Test)
Zdroj: snímek obrazovky Bc. Denise Drdy z DEMO Moodlu

Poté je zapotřebí naplnit test testovými otázkami, neboli úlohami. Po kliknutí na „Přidat“ vyberu ze tří možností. Pro lepší představu přikládám obrázek 2.13 Možnost přidání úlohy. V závislosti na výběru a počtu úloh vložím do testu a následně test uložím a zpřístupním. Následně se zaměřím na popis nejdůležitějších částí. Zvolení možnosti přidá úlohy, následně se objeví v testu požadované úlohy. U jednotlivých úloh je možné vyplnit a rozdělit mezi nimi bodové ohodnocení. Po zmíněných úprav a naplněním testu požadovanými úlohy je nutné test uložit.

Obecná nastavení jedná se o základní nastavení testu, povinné pole název testu a dobrovolné textové pole na popis, kde je vhodné přidat základní informace



Obrázek 2.13: Možnost přidání úlohy
Zdroj: snímek obrazovky Bc. Denise Drdy z DEMO Moodlu

Časování pro každý test je možné nastavit časovou dostupnost v určitém termínu, přes políčka **Zpřístupnit test** a **Uzavřít test**. Dále se nastavuje časový limit pro vyplnění testu. Lze nastavit i chování po vypršení časového limitu.

Známka ve známkování lze nastavit počet pokusů k vypracování testu. Z nabídky lze vybrat z neomezeně nebo od 1 do 10. Způsob vyhodnocení testu je vhodný pro opakované absolvovaní. Zmíněný způsob je dán políčkem **Kategorie známek**.

Následně jej popíši.

Nejvyšší známka do hodnocení testu se v rámci kurzu počítá nejlepší zvládnutý pokus.[8]

Průměrná známka ze všech pokusů se vypočítá průměr a ten se považuje za výsledné hodnocení testu.[8]

První pokus záleží na výsledku prvního pokusu o vypracování testu, ostatní neovlivní celkové hodnocení.[8]

Poslední pokus záleží na výsledku posledního pokusu o vypracování testu, ostatní neovlivní celkové hodnocení.[8]

Rozložení v této části lze nastavit kolik úloh se má na nové stránce zobrazovat.

Chování úloh obsahuje možnost **Zamíchat v rámci úlohy**, to znamená, že je možné promíchávat odpovědi u otázek s více odpovědi. Dále je možné výběrem z **Jak se úlohy chovají** vybrat různá nastavení k efektivnímu využití testu. Pokusím se nyní stručně popíši dle mého názoru nejdůležitější části.

Adaptivní režim případně Adaptivní (bez penalizace) a **Interaktivní s více pokusy** jsou si velmi podobné v tom, že umožňují studentům odesílat odpovědi na úlohy jednotlivě. Po odeslání odpovědi student okamžitě vidí, zda byla odpověď' správná nebo ne a mohou se pokusit o odpovědět na úlohu.[8]

Odložený výsledek umožňuje odesílat odpovědi postupně za jednotlivé úlohy, ale na rozdíl od předchozích režimů nedovolí na úlohu odpovědět opakovaně.[8]

Okamžitý výsledek používá se pravděpodobně nejčastěji. V tomto případě student vyplní celý test, odešle jej a až poté se doví výsledek.[8]

Možnosti prohlídky definuje informace, které budou studentům dos-tupné. Nastavení **V průběhu pokusu** se týká pouze některých testových režimů, např. „Interaktivní s vícero pokusy“, kde se zobrazuje zpětná vazba v průběhu pokusu. Nastavení **Ihned po pokusu o zvládnutí testu** určuje chování v prvních dvou minutách po odeslání pokusu. Nastavení **Později, dokud je test zpřístupněn** upravuje informaci zobrazovanou v období do uzavření testu. Nastavení **Po uzavření testu** určuje, jaká informace se zobrazí studentům po uzavření testu. Pokud není datum uzavření určeno, toto nastavení se nevyužije.[16]

U všech zmíněných prohlídek je možné zobrazovat různá nastavení. Následně je představím.

Pokus odpovědi účastníků testu na jednotlivé úlohy.[8]

Při správné odpovědi odpověď je správná, částečně správná nebo nesprávná.[8]

Body hodnocení za úlohu.[8]

Konkrétní reakce zpětná vazba, podle studentovi odpovědi.[16]

Obecná reakce všeobecná zpětná vazba k úloze.[8]

Správná odpověď správná odpověď na úlohu.[8]

Celková reakce celková zpětná vazba k výsledku testu.[8]

Celková reakce na test obsahuje textové pole na vložení zpětné vazby k celkovému výsledku testu. Vhodné pro popis v rámci mezích hodnocení s komentářem, který se zobrazí po absolvovaní testu.

2.8 Moduly

V kapitole 2.4.1.1 Definice Moodlu jsem již zmínil, že Moodle je modulární systém, který je vzájemně propojený moduly. Konkretně mohou být reprezentovány jako různé typy činností, bloků s aktivity atd. *„Termín modul možná nevysvětluje dostatečně jeho význam. (V anglické verzi Moodlu se používá termín **plugin**).“*[8, strana 319]

2.8.1 Obecně

Pluginy rozdělujeme na **standardní moduly**, jsou obsaženy již od instalace a **rozšíření**, které lze doinstalovat. *„V anglické verzi se pro modul typu **Rozšíření** používá termín **Contributed**, což spíše vystihuje fakt, že jej někdo vytvořil, aby rozšířil možnosti Moodlu.“*[8, strana 320]

2.8.2 Instalace nového modulu

Rozšiřující moduly jsou k dispozici na internetové adrese `https://moodle.org/plugins/`. Než začnu se stažením a instalací je zapotřebí si uvědomit, že u modulu není zaručena funkční správnost.

Podle [8] existuje seznam otázek k doinstalaci modulů

- Skutečně potřebujete modul a jste připraveni ho pravidelně aktualizovat?[8]
- Je modul stále vyvíjen a je k němu poskytována podpora?[8]
- Stáhli jste si modul pro správnou verzi Moodlu?[8]

Celá instalace modulu se skládá z několika kroků. Nejprve je potřeba vyhledat plugin, který chceme instalovat. Pote je potřebné stáhnout ZIP archiv. Následně rozbalit stažený archiv. Dále je nutné postupovat podle přiloženého návodu k instalaci.

2.8.3 Odstranění doinstalovaného modulu

Každý doinstalovaný modul, lze jednoduše smazat. Odstranění je nezvratné, případě navrácení vymazaného modulu je nutné požadovaný modul opětověně naistalovat. Odinstalace probíhá při smazání souborů a vymazání údajů z databáze.

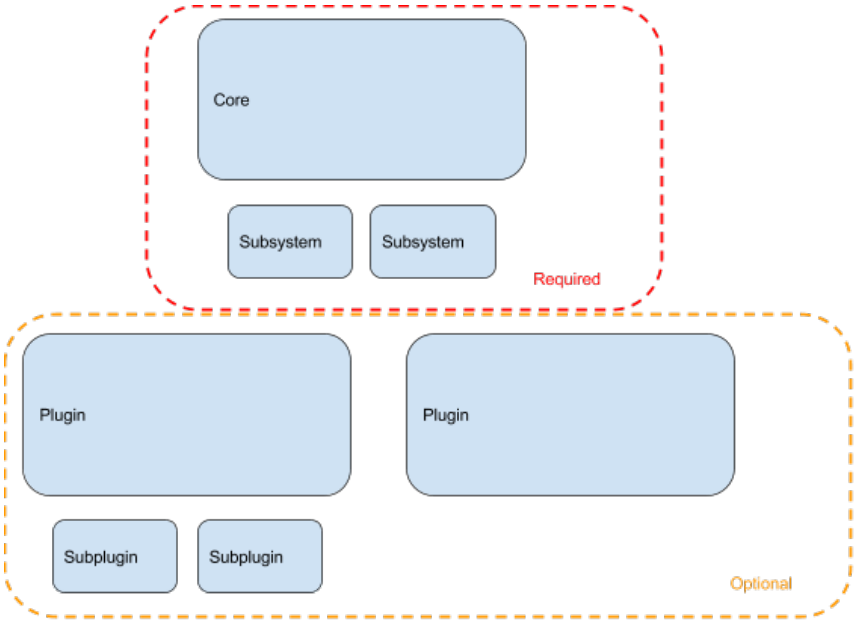
2.9 Moodle pro vývojáře

Moodle je webová aplikace sloužící ke vzdělávání. Má otevřený zdrojový kód v jazyce PHP. Celý systém lze různě upravovat či modifikovat. Dále je možné stahovat různé pluginy, či doknoce vytvářet nové.

V následující kapitolách popíši struktury a principy chování programování.

2.9.1 Komponenty

Celý systém Moodle je rozdělen do různých sekcí. Zmíněné sekce se nazývají komponenty. Pro lepší znázornění přikládám obrázek 2.14 Komponenty v Moodlu. Následně termíny stručně popíši.



Obrázek 2.14: Komponenty v Moodlu
Zdroj: `https://docs.moodle.org/dev/Communication_Between_Components`

2.9.1.1 Jádro

Jedná se o úplný základ pro fungovaní systému. Tato komponenta se nazývá jádro a obsahuje základní knihovny. Knihovny obsahují základní funkce. *„Základní kód není volitelný a nelze jej bezpečně odstranit bez porušení Moodle. To znamená, že je vždy k dispozici a lze jej bezpečně zavolat odkudkoli.“*[17, Hlavní strana ▶ Komunikace mezi komponenty]

2.9.1.2 Subsystém

Jsou to skupiny součástí jádra a logicky seskupeny dohromady. Tyto skupiny obsahují funkce a třídy pro fungování jádra. Často jsou závislé na dané funkci. Většinou je lze nastavit či dokonce deaktivovat pomocí konfiguračního systému, kde není vidět kód. Bez subsystému by systém Moodle nefungoval.

2.9.1.3 Plugin

Pluginy neboli moduly jsou volitelné součásti systému. Umožňují rozšířit i jeho funkčnost. Jak jsem dříve zjistil, „M“ v názvu Moodle znamená modulární a proto většina kódu v Moodle patří pluginům. Existují různé typy modulů pro různorodé rozšíření základních funkcí Moodlu.

2.9.1.4 Typy pluginů

Balíček Vanilla Moodle[17] obsahuje přes 370 pluginů. Dále je možné další moduly stáhnout z adresáře *Plugins*. Také je možné nainstalovat ručně, jak jsem již zmínil v kapitole 2.8.2 Instalace nového modulu.

2.9.1.5 Dílčí pluginy

Některé typy pluginů umí podporovat i zásuvné pluginy neboli moduly. Jedná se o pluginy, které lze rozšířit dalšími pluginami. *„Jediné typy pluginů v Moodle, které to umožňují, jsou moduly aktivit, editory, nástroje pro správu a lokální pluginy.“*
[17, Hlavní strana ▶ Komunikace mezi komponenty]

2.9.2 Komunikační kanály

Komunikace mezi kanály je způsob volení kódu v Moodlu. Například lze volat kód:

- Přímé volání funkcí PHP
- Externí funkce
- Moduly Javascript (AMD)
- Šablony
- Volání get_string
- Pozorovatelé událostí
- Zpětná volání komponent

2.9.3 Architektura

Jedná se o fungování Moodlu po technické stránce. Celá architektura slouží k provozu systému. Moodle je napsán v jazyce PHP.

V této části představím některé důležité části konceptu architektury. Jedná se o Modulární systém, databáze, operační systém atd.

2.9.3.1 Modulární systém

V kapitole 2.4.1.1 Definice Moodlu jsem vysvětlil zkratku Moodlu, který se skládá z prvních písmen výrazů. Z pohledu programování mě bude zajímat „M“ z názvu. Pro připomenutí se jedná o modularitu. *„Moodle je navržen tak, aby byl vysoce rozšiřitelný a přizpůsobitelný bez úpravy základních knihoven, protože by to způsobilo problémy při upgradu Moodle na novější verzi.“*[17, Hlavní stránka ▶ Kódování ▶ Architektura Moodle]

Z těchto důvodů systém umožňuje programátorovi vytvářet různé úpra-vy, či moduly bez velkých zásahů do celého kódu. Tento způsob úpravy je velmi

vhodný, neboť umožňuje zkrátit dobu k provedení úprav. A jak všichni víme: „čas jsou peníze.“

2.9.3.2 Databáze

„Databázová vrstva Moodlu je napsána pomocí knihovny PHP ADOdb, která byla vytvořena poskytnout standardizovaný způsob přístupu k různým databázovým systémům pomocí a konzistentní programovací rozhraní.“[18, strana 30]

Obsahuje speciální nativní databáze knihovny PHP, sloužící k fungování programu, které podporují více databázových serverů. Knihovna ADOdb poskytuje Moodle podporu pro různé databáze: MySQL, PostgreSQL, Microsoft SQL a Oracle.

2.9.3.3 Operační systém

Prostředí Moodle lze spustit na libovolném operačním systému. Pouze je nutná podpora jazyku PHP a databáze v závislosti na verzi.
(Popřípadě je nutné na tvorbu požadované verze doinstalovat.)

2.9.3.4 Webový server

Moodle lze nahrát na jakýkoliv webový server, který podporuje potřebnou verzi PHP. Nejpoužívanější webový server je Apache. Jeho výhodou je široká dostupnost na všech operačních systémech.

Moodle je webová aplikace s omezením podpory a vývoje. V tom je celý princip Moodlu a hlavně jeho otevřenost pro vývoj a různorodá přizpůsobení. *„Uživatel interakce s Moodle bude mít vyšší než normální počet kliknutí a Moodle generuje mnoho SQL dotazů při vytváření stránky. Moodle je velmi efektivní dělá. To, co dělá, ale je vše poměrně složité. To znamená, že jako vývojáři potřebujeme abychom si byli vědomi typu architektur, které budou naše modifikace pravděpodobně použity v rámci. Znamená to také, že si musíme být vědomi důsledků našeho výkonu kódování.“*[18, strana 32]

2.9.3.5 Adresářová a systémová struktura

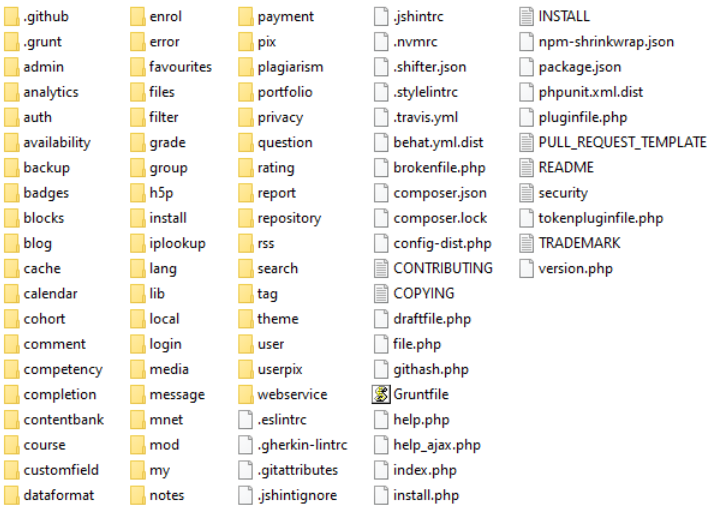
Celý systém Moodlu se dělí na tři samostatné oblasti: kód Moodle, databáze Moodle a data Moodle. Tyto části stručně definuji v následující podčásti.

Moodle kód Moodle je vytvořený v interpretovaným jazyce PHP. To znamená, že systém je uložen jako zdrojové kódy v souborech na webovém serveru. Na server je nahrán konkrétní PHP soubor. Následně je kód serverem analyzován a výsledný výstup z kódu je zobrazen uživateli. Každý soubor je umístěn v příslušném adresáři a tím vznikla adresářová struktura. V tom je princip již zmíněného termínu „Modulární“.

Každá složka na nejvyšší úrovni je hlavní součástí Moodlu. Mnoho hlavních komponent podporuje zásuvné moduly. Každý plugin má vlastní složku uvnitř

složky komponenty.[18] Mohou i podporovat další moduly, např. modul kvízových aktivit, který podporuje další moduly. Koncový uživatel tuto magii nevidí., protože moduly jsou nainstalovány a následně zkopírovány do příslušné složky na serveru.

Pro lepší zobrazení přikládám obrázek 2.15 Seznam adresářů v Moodlu. Představím pouze základní a důležité složky. Moodle má jednoduchá pravidla pro tvorbu modulů. Všechny moduly jsou ve své vlastní složce a název složky je název, který Moodle zobrazí ve svém rozhraní při odkazu na modul.



Obrázek 2.15: Seznam adresářů v Moodlu
Zdroj: snímek obrazovky Bc. Denise Drdy z adresáře Moodlu

Admin v této složce jsou uloženy soubory PHP, které ovládají rozhraní administrátora.[17]

Auth obsahuje všechny autentizační moduly pro Moodle. Každý modul bude mít v této oblasti svůj vlastní adresář.[18]

Backup tato složka obsahuje základní prostředky pro zálohování kurzu pro systém.[17]

Blocks složka bloky se používá k zobrazení polí s informacemi na pravé nebo levé straně sloupce na straně stránky Moodle.[18]

Course tato složka Moodle má zřejmý význam vzhledem k tomu, že Moodle je organizován kolem kurzů.[18]

Enrol složka pro zápis obsahuje všechny moduly pro zápis do Moodlu.[18]

Files komponenta soubory umožňuje Moodlu začlenit soubory do systému.[17]

Filter systém filtrování Moodle je systém vyhledávání a nahrazování, založený na textu regulárních výrazech zařízení.[18]

Lang standardní Moodle je dodáván pouze s anglickým jazykovým balíčkem. Anglické termíny (neboli 'řetězce') pro hlavní (základní) funkce Moodlu, jako je administrace, role atd., jsou uloženy v lang/en[17]

Lib složka lib uchovává funkce základní systémové knihovny.[18]

Mod složka mod ukládá moduly aktivit, jako je úkol, kvíz, wiki, fórum a moduly lekcí.[18]

My je lehká portálová oblast v Moodle. Poskytuje seznam kurzů, které jsou studentovi přiděleny, včetně shrnutí nadcházejících aktivit kurzu.[17]

Thema ve složce témat jsou uloženy všechny vestavěné motivy Moodle a vlastní témata nainstalovaný v systému. Motivy jsou kombinací CSS, HTML a PHP.[18]

Databáze Moodle celá databáze Moodlu je uspořádána přibližně do 200 propojených tabulek. *„Každá hlavní komponenta systému má obvykle jednu nebo více tabulek, z nichž každá začíná názvem komponenty.“*[18, strana 38] Mezi důležité požadavky patří zacházet s celou databází jako s Entitou, včetně kopírování a přesouvání instancí celé databáze Moodle pro vytváření pracovních a testovacích oblastí při vývoji. To celé se provádí nejčastěji pomocí příkazových řádků

Data Moodle slouží jako úložiště souborů pro uživatelsky nahraný obsah. Tyto data ukládají i informace o uživatelých včetně jejich přihlášení do systému. Dále se ukládá vybraný jazyk z jazykových balíčků.

Většina dat a soubory jsou ve složek podle kurzu. Každý kurz má složku, která je pojmenována unikátní celočíselnou hodnotou. Hodnota je nastavena na ID interní databáze daného kurzu.

2.9.4 Základní rozhraní API

především se jedná o řadu základních API, které poskytují různé nástroje pro Moodle. Tyto skripty jsou potřebné k psaní přídavných modulů. Následně představím nejpoužívanější API rozhraní. Jako poslední uvedu API pro tvorbu otázek a čerpání dat z banky úloh.

Většina hlavních knihoven Moodlu je uložena v adresáři */lib/*[17].
Všechny tyto soubory dodržují pravidla pojmenování *[funkce] lib.php*[18]. Tyto soubory knihovny obsahují Moodle API.

Access API (přístup) poskytuje funkce, takže si můžete určit, co je aktuální uživatel nemá dělat, a to umožňuje moduly rozšířit Moodle s novými schopnostmi.[17]

API pro manipulaci s daty (dml) umožňuje číst/zapisovat do databází konzistentním a bezpečným způsobem.[17]

File API (soubory) řídí ukládání souborů ve spojení s různými pluginy.
[17]

Form API (formulář) definuje využití uživatelských dat prostřednictvím webových formulářů.[17]

Event API (událost) protokoluje události v Moodle, zatímco protokolování 2 popisuje, jak jsou protokoly uloženy a vyvolány.[17]

Navigation API (navigace) manipuluje navigační strom přidáváním a odstraňováním položky tak jak si přejete.[17]

API page (stránka) se používá k nastavení aktuální stránky, přidání JavaScriptu a konfiguraci toho, jak se věci budou zobrazovat uživateli.[17]

Output API (výstup) slouží k vykreslení HTML pro všechny části stránky.[17]

String API (řetězec) ukazuje jak použít jazyk textových řetězců v uživatelském rozhraní. Zvládá všechny dostupné jazykové překlady.[17]

Upgrade API (upgrade) instaluje modul a sám jej upgrady. (Od sledování vlastní verzí.)[17]

Moodlelib API (jádro) ústřední knihovna je soubor různorodých univerzálních funkcí Moodle. Funkce mohou ovládat parametry požadavku, konfigurace, uživatelské preference, čas, přihlášení, mnet, pluginy, řetězce a další. Existuje také spousta definovaných konstant.[17]

Question API (otázka) ty lze rozdělit na bankovní API otázky a API otázka enginu, mohou být použity činnosti, které chcete použít na otázky od otázky banky.[17]

2.10 Programování

Podle dokumentace[17] slouží jako návod s pokyny pro kódování do Moodlu. Jedná se o rozložení textu a příkazů v kódu. Dále slouží k sjednocení způsobu úpravy a formátování PHP kódu, které je vhodné pro vývojové prostředí projektu s více programátory.

Abstraktní cíle, o které usilujeme:

jednoduchost čitelnost přívětivost nástrojů, jako je použití signatur metod, konstant a vzorů, které podporují nástroje IDE a automatické doplňování názvů metod, tříd a konstant.

„Abstraktní cíle, o které Moodle usiluje:

- *jednoduchost*
- *čitelnost*
- *přívětivost nástrojů, jako je použití signatur metod, konstant a vzorů, které podporují nástroje IDE a automatické doplňování názvů metod, tříd a konstant.*“[17, Hlavní strana ▶ Kódování ▶ Styl kódování]

2.10.1 Styl kódování

„Pokud není uvedeno jinak, bude tento dokument kódování podřízen PSR-12 a PSR-1 v tomto pořadí.“[17, Hlavní strana ▶ Kódování ▶ Styl kódování]

V případě velkých či sdílených projektech je vhodné používat jednotný styl programování. V následujících podkapitolách se zaměřím na stručné informace o zmíněných standardech.

2.10.1.1 PSR-1: Základní kódovací standard

Základní část standardu obsahuje základní pravidla pro sdílený PHP kód.

„Přehled:

- *Soubory MUSÍ používat pouze* `<?php` *a* `<?=` *tagy.*
- *Soubory MUSÍ používat pouze UTF-8 bez kusovníku pro kód PHP.*
- *Soubory BY MĚLY bud’ deklarovat symboly (třídy, funkce, konstanty atd.) nebo způsobovat vedlejší efekty (např. generovat výstup, měnit nastavení .ini atd.), ale NEMĚLY by dělat obojí.*
- *Jmenné prostory a třídy MUSÍ následovat po "automatickém načítání"PSR: [PSR-0 , PSR-4].*
- *Názvy tříd MUSÍ být deklarovány v* `StudlyCaps`*.*
- *Konstanty třídy MUSÍ být deklarovány velkými písmeny s oddělovači podtržítka.*
- *Názvy metod MUSÍ být deklarovány v* `camelCase`*.*“[19, PSR-1]

2.10.1.2 PSR-12: Rozšířený styl kódování

Jedná se o rozšíření a nahrazení předchozího standardu PSR-2. Rozšířený styl musí dodržovat pravidla ze základního standardu PSR-1.

„Tento PSR se snaží poskytnout stanovený způsob, jak mohou nástroje stylu kódování implementovat, projekty mohou deklarovat dodržování a vývojáři se mohou snadno používat mezi různými projekty. Když různí autoři spolupracují na více projektech, pomáhá mít jeden soubor pokynů, který lze použít pro všechny tyto projekty. Výhoda této příručky tedy nespočívá v samotných pravidlech, ale ve sdílení těchto pravidel.“[19, PSR-12]

2.10.1.3 Nástroje

Existuje řada různých nástrojů, která se používá při psaní kódu. V Moodlu se používají konkrétně dva nástroje. Využití je především ke kontrole kódu. Podle dokumentace[17] je vhodné pracovat s oběma, protože oba provádějí mírně odlišné kontroly.

Kontrola kódu
`https://moodle.org/plugins/view.php?plugin=local_codechecker`

Kontrola PHPdoc Moodle
`https://moodle.org/plugins/local_moodlecheck`

2.10.2 Programovací pravidla

Většinou v projektech je dán standardní styl programování.

„Napsání kódu tímto způsobem je důležitým krokem k tomu, aby byl váš kód akceptován komunitou Moodle.“[17, Hlavní strana ▶ Kódování]

V následující části se zaměřím vybraným pravidlům pro styl kódování.

2.10.2.1 Formátování souboru

Nyní se budu věnovat vybraným pravidlům na formátování souboru.

PHP tagy používá se tzv. „dlouhé“ značky PHP. Kvůli problémům s mezerami.

Listing 2.1: PHP tagy[17]

```
<?php

require('config.php');
```

Odsazení vždy se používá odsazení o **4 mezerách** bez tabulátoru. Z důvodu zabránění vkládání nových znaků tabulátoru do zdrojového kódu. Neodsazuje se hlavní úroveň skriptu.

Listing 2.2: Odsazení[17]

```
<?php
require('config.php');
$a~= required_param('a', PARAM_INT);
if ($a~> 10) {
    call_some_error($a);
} else {
    do_something_with($a);
}
```

Maximální délka řádku z důvodu čitelnosti je omezená na pevnou maximální délku. Doporučená délka je 132 znaků. Nedoporučuje se používat více než 180 znaků.

2.10.2.2 Pravidla pojmenování

nyní se budu zabývat vybraným pravidly, která mají svojí konvenci pojmenování.

Názvy souborů musí být celá anglická slova. Slovo by mělo být co nejkratší a psané pouze malými písmeny. Pojmenování končí na přípony souborů .php, .html, .js, .css nebo .xml.

Třídy názvy tříd jsou vždy anglická slova s malými písmeny, oddělená podtržítky

Listing 2.3: Třídy[17]

```
class  some_custom_class  {
    function  class_method ()  {
        echo  'foo' ;
    }
}
```

Proměnné názvy by měly být v angličtině. Především snadno čitelné a smysluplné s malými písmeny. Pokud je potřeba více než jedno slovo, tak se napíše bez mezer hned za sebe. Používat názvy v množném čísle pro pole objektů. Vždy používat kladné názvy proměnných (povolit, povolit, nezabránit, zakázat).

2.10.3 Bezpečnost

bezpečnost slouží k ochraně zájmů všech uživatelů. Na Moodlu se objevují citlivá a důležitá data. Jako jsou soukromé diskuze či známky před širokou veřejností zvenčí. Jména a emailové adresy jsou také skryté.

„Obecná doporučení:

- *samostatný administrační backend*
- *ve webové aplikaci nejsou uloženy žádné citlivé informace*
- *komunikace musí být šifrována pomocí SSL*
- *musí se zaznamenat všechny akce uživatele*
- *serverové aplikace musí být zcela odděleny*
- *nemohou existovat žádné soubory nahrané uživateli na server*
- *neexistuje žádný formátovaný text zadaný uživateli na serveru (omezený pouze prostý text)*
- *musí se ověřit identita uživatele a akce prostřednictvím samostatného kanálu*
- *vždy udržet každý software aktuální*

- *nedoporučují se žádná rozšíření prohlížeče třetích stran*
- *je důležité používání pouze jedné webové stránky, neotevírání více oken s různými stránkami, zavření/otevírání prohlížeč před a po použití zabezpečené*“[17, Hlavní strana ▶ Kódování ▶ Bezpečnost]

2.10.4 XHTMl, CSS a JavaScript

Moodle vyžaduje stritkní a dobře strukturovaný kód HTML5. V případě možnosti se zpětnou kompatibilitou s XHTML1.1. Na rozvržení se používá CSS. Moodle v základu obsahuje několik nainstalovaných motivů. Základní nastavení je vhodné pro uživatele s výchozími motivy, které lze pak podle nastavení různě barevně upravovat. Např. tmavý režim.

JavaScript pro Moodle vyžaduje strukturu podle Vanilla JavaScript ve stylu ES6[17]. Používání dalších frameworků se nedoporučuje, kvůli odlišnému přístupu do jádra. Obecně by měl být kód napsán tak, aby se zabránilo zobrazování rozhraní, která jsou odstraněna, nebo přidávání nových rozhraní při načítání stránky. Jelikož se jedná o otevřený zdrojový kód, tak veškerý Javascript musí být přístupný.

2.10.5 Jazykové prostředí

Celý Moodle funguje ve více než 100 jazykových překladů. Ale zdrojové kódy jsou odděleny od jazykových řetězců. Tím se opět jedná od modulární systém. Výchozím jazykem pro veškerý kód, komentáře a dokumentaci je angličtina.

2.10.6 Testování

Testování probíhá již během integrace, tak i testovacím týmem Moodle. Všechny problémy jsou integrované do základní kódové základny Velká část testování je automatizovaná, existuje mnoho částí, které automatizovat nelze a je vyžadováno ruční testování.

2.11 Pravidla pro přispívající kód

Sdílení a zveřejnění vytvořeného pluginu s komunitou. Ale není to nutností, když to daná situace nevyžaduje. Jedná se o situaci, kdy plugin řeší potřeby daného webu anebo vytvořený modul nechci sdílet s ostatními.

2.11.1 Výhody registrace

V registraci modulu v adresáři *plugins* má řadu výhod. Instalace vytvořeného pluginu je jednoduší. Stačí pouze Správce, který nainstaluje ze zmíněného adresáře. V případě aktualizace Správci dostanou upozornění. Pomocí uložení vytvořeného pluginu si vyhrazuji název modulu.

V poslední řadě musí být vytvořený modul schválen a mnohdy je problém s instalací. Mnoho stránek Moodlu je nastavená, tak že respektuje kontroly schvalování a nepovoluje instalaci neschváleného pluginu na své servery.

2.11.2 Struktura

Jsou daná pravidla na sdílení vlastního modulu pro ostatní členy Moodle komunity. Než se vytvořené dílo odešle do adresáře *plugins*, tak by měl obsahovat požadované náležitosti. Následně se jim budu stručně věnovat

Typ pluginu jsou určené podle typy funkce. Pro implementaci určité funkce je zapotřebí vybrat vhodní typ modulu.

Název pluginu je jedinečné pojmenování a slouží k identifikaci modulu Moodle na základě jeho názvu a typu pluginu.

Úložiště místo kde se daný kód nachází. Měl by být veřejně dostupný, aby mohlo dojít k případným úpravám a dalšímu vývoji.

Tracker je potřebný pro hlášení chyb, námětů a připomínek na funkce pluginu.

Dokumentace by měla obsahovat všechny požadované náležitosti.

Snímky obrazovky zobrazují základní funkce vytvořeného pluginu.

2.11.3 Postup sdílení kódu

Celý postup je podle dokumentace[17]. Nyní se budu věnovat jednotlivým krokům.

Nahrát první počáteční verze pluginu ke schválení. Pro snadné schválení je zapotřebí řídit se pokyny pro přispívání pluginů.

Odeslání ke schválení trvá většinou v řádů týdnů, než dostanu zpětnou vazbu a zprávy z výsledné kontroly.

Schvalování prochází procesem ověření a schválení modulu.

Zpětné vrácení většina pluginů neprojde počáteční kontrolou. (Potřebují doprogramovat či opravit chyby.)

Schválení modul je zároveň registrován u AMOSu a lze vytvořit jeho překlad. Samotný plugin by měl být dodáván pouze s anglickými řetězci.

Aktualizace probíhá pouze prostřednictvím adresáře *plugins*.

Kapitola 3

Praktická část

Nejprve se zaměřím na stažení a instalaci aplikace Moodle na vlastním serveru. Ručně přidám testovou úlohu, abych věděl složitost přidání do systému Moodle. Podrobně se budu věnovat vývoji a implementaci nového prototypu pluginu pro zlepšení zadávání speciálních algoritmů. Ve zmíněné podkapitole se zaměřím na analýzu budoucího pluginu a návrh řešení. Dále naimplementuji a shrnu detailním popisem vytvořený plugin.

V další kapitolu věnuji Návodu. Kde se zaměřím na jeho instalaci a uživatelskému použití. Jak ze strany tvůrce testu, tak i ze strany studenta. Bude se podrobně věnovat možnému nastavení a zadání testu. Následně provedu testování vytvořeného prototypu.

3.1 Instalace Moodlu

Na vývoj přídavného modulu je zapotřebí mít vlastní instalaci programu Moodle. Jedná se o nainstalování Moodlu na svůj vlastní server. Je mnoho způsobů jak nainstalovat Moodle. K samotné aplikaci Moodle je potřeba další potřebné aplikace.

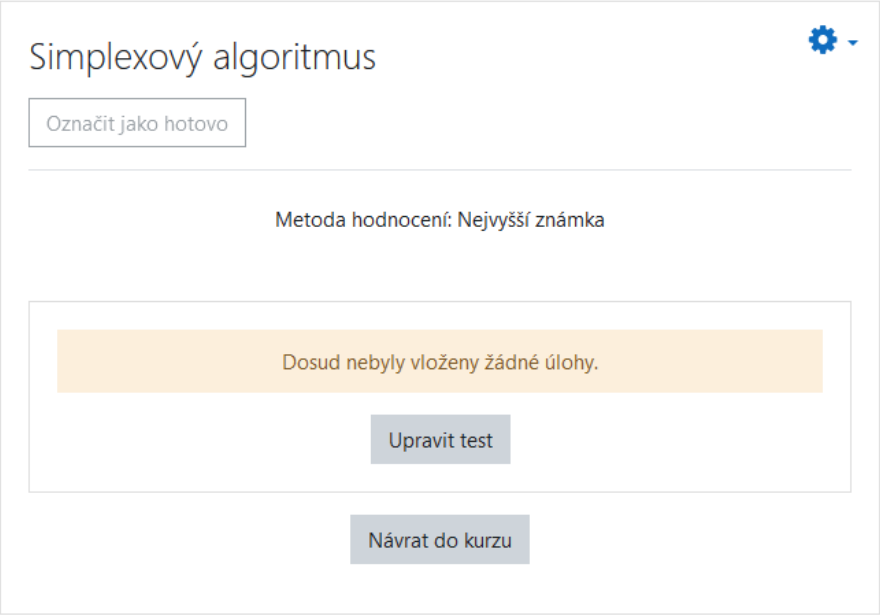
Zdá se mi nejsnazší způsob stažení kompletního all-in-one instalačního balíčku z `http://moodle.org` v sekci Stahovaní. Jsou rozděleny podle operačního systému. Balíček typu all-in-one v překladu znamená „vše v jednom“, proto obsahuje nejen Moodle, ale i ostatní nezbytný serverový software, včetně Apache, PHP a MySQL. Podle [18] se zmíněné balíčky nedoporučují pro produkční použití, mohou poskytnout pohodlný výchozí bod pro začínající vývojáře. Tento způsob je vhodný zejména pro zkoušení a hraní si s novými moduly, před nahrátí pluginu do ostré provozní verze.

Pro vývoj celého nového pluginu je nutné zprovoznit kompletní instalaci Moodlu. Lze volně stáhnout z `http://moodle.org` v sekci Stahovaní. Instalace vyžaduje nainstalování XAMP, PHP a MariaDB nebo MySQL databáze. Poté lze vytvořit vlastní server s instalací Moodle, kde se dokončí instalace pomocí průvodce. Jedná se o postupný průchod nastavení a konfigurace prostředí serveru, včetně potvrzení lincenčních podmínek. V posledním krokem je nastavení správce kurzu a následně vytvoření samotného kurzu.

3.2 Ruční přidání testu na Simplexový algoritmus

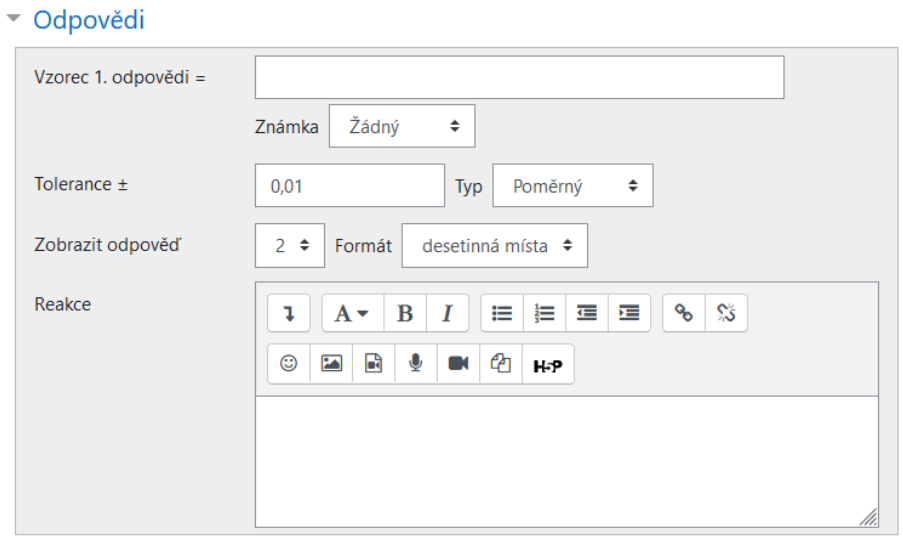
Podle mého názor je v originální verzi Moodlu, bez dalších pluginů nejvhodnější vyžití na testování Simplexového algoritmu použít Vypočítávanou úlohu. Protože při testu je možné vybrat z datové sady různé proměnné pro odlišné varianty testu. Následně se automaticky vyhodnotí a porovná se zadaným výsledkem od studenta. Student vidí okamžitě výsledek.

Nejprve přidám činnost, ze které vyberu Test. Následně otevřu vytvoření test. Nový test je prázdný, bez úloh, jak je vidět na obrázku 3.1 Vytvořený nový test Moodlu.



Obrázek 3.1: Vytvořený nový test Moodlu
Zdroj: snímek obrazovky Bc. Denise Drdy z Moodle

Dále je zapotřebí přidat konkrétní úlohu přes úpravu testu. Pro Simplexový algoritmus jsem zvolil Vypočítávanou úlohu. Vyplním potřebné políčka úlohy. Místo čísel volím proměnné pro lepší univerzálnost, neboť se následně doplní z datové sady. Problém nastává, až u odpovědi, kde jde vložit pouze jeden vzorec pro lepší představu přikládám obrázek 3.2 Vložení úlohy na Simplexový algoritmus v Moodlu.



Obrázek 3.2: Vložení úlohy na Simplexový algoritmus v Moodlu
Zdroj: snímek obrazovky Bc. Denise Drdy z Moodle

Bohužel nejde kombinovat několik vzorců a vytvářet cykli na počítání. Zmíněný plugin umožňuje pouze výpočet jednoho vzorce, abych vytvořil Simplexovou úlohu je nevyhovující. Proto usuzuji, že jsem se mýlil a pro tento typ úlohy je nevhodné použít algoritmus založení na rozhodování a složitém počítání.

Z těchto uvedených důvodů vyvinu nový prototyp na zadávání simplexového algoritmu do Moodlu. Bude součástí testových úloh. Více podrobností bude v následujících kapitolách.

3.3 Vývoj nového protypu pluginu

V této rozsáhlé části se zaměřím na celou implementaci nového prototypového pluginu. Nejprve začnu s analýzou možné tvorby modulu, včetně vyhodnocení požadavků. Dále využiji a popíšu doporučovanou šablonu, používající se na naprogramování testové otázky. V další části se zaměřím na návrh, který bude vycházet z vytvořené analýzy. Následně naimplementuji a detailně popíši vytvořený plugin.

3.3.1 Analýza řešení

V této podkapitole se budu rovněž zabývat analýzou, včetně zohlednění požadavků na modul. Následně návrh řešení pluginu na zadávání speciálních algoritmů.

3.3.1.1 Požadavky na modul

K hlavním požadavků patří zadávání speciálních algoritmů do systému Moodle na testování studentů. Zaměřím na problém lineárního programování. Konkrétně na testování Simplexového algoritmu, kterému jsem se věnoval v teoretické části. Dále musí plugin umět počítat s různými počty proměnnými a omezujících podmínek, tak aby mohl vzniknout odlišné příklady. Poslední zmíněný požadavek by měl zamezit opisování mezi studenty. Samozřejmostí je automatické vyhodnocení dané otázky tak, aby student viděl okamžitě výsledek své práce.

3.3.1.2 Analýza

Požadavek vytváření testů na zadávání a testování Simplexového algoritmu, nelze uskutečnit ve standardní verzi Moodlu. Ani neexistují dodatečné pluginy, které by zvládly složité výpočty zmíněného problému. Takže v zásadě není možné, jak docílit vytváření testových úloh pro lineární programování.

Proto je nutné vytvořit vlastní plugin pro vytváření a hodnocení testů. Jedná se o vytvoření modulu typu otázka (v angličtině question). Přidat jej mezi testové otázky a vytvořit jej snadno dostupný. Musí být jednoduše ovládatelným rozhraním pro učitele tak, aby snadno mohl uživatel snadno a přehledně zadat test. Umožnit učiteli po zadání vstupních informací vidět správný výsledek pro kontrolu ve správnosti hodnot.

V rámci implementace výpočtu Simplexové úlohy je nejefektivnější využít externí knihovnu na výpočet Simplexového algortimu v PHP od tvůrce Petra Kesslera [20], než vytvářet vlastní rozsáhlou knihovnu na výpočet. Celá knihovna je vydaná pod licencí MIT. Zmíněnou licenci jsem se zabýval již ve své bakalářské práci: *„MIT lze použít jako proprietární software nebo GPL licencovaný software při zachování textu podmínek.“*
[9, strana 61]

V poslední řadě je zapotřebí implementovat i plugin pro zadávání testové otázky. Správné zakomponování knihovny do pluginu a zároveň nejoptimálnější propojení pro výpočet optimálního řešení. Dále naprogramování dynamického

zobrazování zadávaní testu a zobrazování testové otázky pro studenty, včetně nastavení hodnocení. Závěrem je nutné vytvořit českou a anglickou lokaci.

3.3.1.3 Uspořádání modulu

Po předchozí analýze možného řešení se vhodné řešení nabízí využít modul pro testovou otázku. Zároveň bude nový plugin zahrnut mezi kvízové otázky a ukládat se do banky úloh. To znamená, že vytvářený plugin bude umístěn a vyvíjen v příslušném adresáři pro testové otázky, konktrétně ve složce *qtype*. Tímto bude splněn požadavek na snadnou dostupnost pro učitele.

Celá realizace prototypového pluginu bude prováděna na základě předchozí analýzy. Zároveň musí vycházet z programovacích praktik a požadavků samotného systému Moodlu. Pro vývoj bude nutné navrhnout databázovou strukturu pro vyvíjený plugin v závislosti na další modularitě.

Pro snadnější modifikaci a následnou urdžitelnost celého pluginu je zapotřebí vyvíjet modul na poslední verzi systému Moodlu. Proto jsem se rozhodl pro programování v poměrně nové verzi 3.11.4. Musím dodat, že celý systém se stále vyvíjí. Staré verze Moodlu umožňují odlišné programovací principy. Mají různé druhy kompatibilit v každé verzi. Z toho důvodu je nejdůležitější pracovat s příslušnou dokumentací.

3.3.1.4 Využití a hodnocení testů

Využití je především vhodné pro učitele zabývající se zmíněnou problematikou Simplexového algoritmu a ulehčení vkládání a hodnocení testů. Zároveň z toho vyplývá umožnění testování studentů v tomto problému. S tím souvisí i hodnocení testů, který jsem se rozhodl, že musí být hodnocen automaticky podle pravidel Moodlu. Nejen automatický výpočet jako kontrolu v zadání učitele, ale i zobrazení okamžitě výsledek pro studenty.

3.3.2 Šablona

Podle vývojářské dokumentace[17] je vhodné použít šablonu (template) ke tvorbě otázky na naprogramování pluginu. Jedná se o prázdnou kostru modulu, který je slouží pro začátek implementace vlastního typu otázky v systému Moodlu. Šablona vznikla kopií otázky typu krátká odpověď a bylo z ní vymazáno vše nepotřebné. Jako je zpětné třídění, rady, text otázky, hodnocení, vstupní ovládací prvky a databázové tabulky [21].

3.3.2.1 Použití

Sám autor šablony James Pratt uvádí: *„Toto je jeden alternativní začátek pro vývoj zásuvného modulu typu otázka a funguje kód tak, jak je. Ačkoli neprovádí žádné skutečné hodnocení ani neshromažďuje vstupní informace od studentů.“* [21, jamiepratt/moodle-qtype_TEMPLATE]

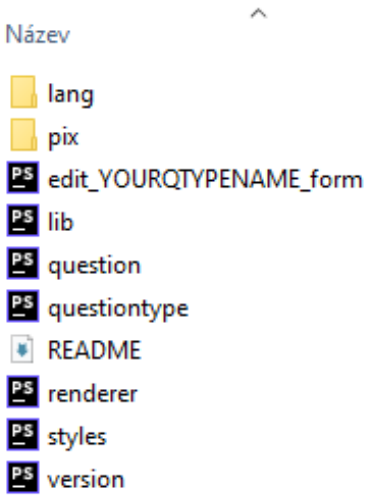
Z těchto uvedených důvodů existují dvě varianty vývoje nového pluginu. Celý výběr variant je čistě na programátorovi. Existují různé výhody a nevýhody každého výběru, ale podle mého názoru záleží na vhodnosti použití pluginu a zkušenostech vývojáře.

Použití stažené šablony ve vývojářské dokumentaci Moodlu [17] je možné bezplatně stáhnout základní šablonu na tvorbu nového modulu. Ale musím upozornit, že existuje mnoho šablon v závislosti na verzi Moodlu. Také několik dokumentací ke každé verzi. Mnohdy nejsou aktualizované a odkazují na nejnovější šablonu, která je na velmi starou verzi Moodlu. Doporučuje se použít šablonu pro verzi, ve které se chystám programovat. Většinou to bývá poslední verze nebo rovnou se vyvíjí ve stabilní vývojové verzi, aby náš program byl aktuální.

Zkopírování a úprava exitsujícího modulu nejprve si musím najít nejvhodnější plugin v systému Moodle. Podle dokumentace [17] se doporučuje použít oficiální modul ze základní instalace. Poté celou složku nakopíruji, vymyslím jméno pro svůj plugin a vymažu nepotřebné části kódu. Následně upravím záhlaví v jednotlivých souborech podle požadované struktury v Moodlu. Tímto postupem vytvořím také šablonu vhodnou pro programování.

3.3.2.2 Adresářová struktura šablony

Pro lepší představu o struktuře šablony přikládám obrázek 3.3 Seznam adresářů šablony. Jedná se o výpis adresářů ze stažené šablony. Následně představím důležité a používané složky pro vývoj nového pluginu.



Obrázek 3.3: Seznam adresářů šablony
Zdroj: snímek obrazovky Bc. Denise Drdy z adresáře šablony

edit_YOURQTYPENAME_form.php používá se k definování formuláře pro úpravu otázky.[17]

questiontype.php definuje třídu qtype_mytype . Toto musí rozšířit question_type.[17]

question.php obsahuje definici třídy qtype_mytype_question, která by měla rozšířit základní třídu question_definition.[17]

renderer.php obsahuje definici třídy qtype_mytype_renderer, která by měla rozšířit základní třídu qtype_renderer.[17] Tato třída slouží ke zobrazování testové otázky pro studenty.

lang/en/qtype_myqtype.php anglické jazykové prostředí. Kde se zástupný znaky, či řetězce definují jako texty, jména,...

version.php informace o verzi pro typ otázky. Také zde můžete zahrnout, kterou verzi základního Moodlu a/nebo jiné typy otázek tento modul vyžaduje.[17]

3.3.2.3 Instalace šablony

Nejprve je nutné soubor ve formátu ZIP stáhnout z GitHubu od tvůrce šablony [21]. Samozřejmě nejnovější verzi a následně rozbalit do adresáře Moodlu konkrétně do složky Question/type. Poté je zapotřebí nově vytvořený adresář přejmenovat z TEMPLATE na vlastní identifikátor. Pokud se ponechá název TEMPLATE, bude zásuvný modle ignorován Moodlem a neobjeví se mám v našem prostředí.

3.3.2.4 Aplikace šablony

Mezi důležité kroky patří následující rady:

- *„Zkopírujte nebo přejmenujte adresář modulu na YOURQTYPENAME.*
- *Nahrad'te všechny výskyty YOURQTYPENAME v souborech novým názvem vašeho typu otázky.*
- *Přejmenujte soubory, které mají YOURQTYPENAME nahrazující YOURQTYPENAME novým názvem pro váš typ otázky.*
- *Nahrad'te „@copyright THEYEAR YOURNAME (VAŠE KONTAKTNÍ INFORMACE)" něco jako @copyright 2013 Jamie Pratt (me@jamiep.org)"* [21, jamiepratt/moodle-qtype_TEMPLATE ▶ Použití]

3.3.3 Návrh

Návrh se bude řídit předchozí analýzou. Jak jsem se již zmínil, bude se celý prototyp pluginu zabývat Simplexového algoritmu, včetně jeho výpočtu i kontroly správného řešení.

3.3.3.1 Návrh pojmenování

Pro Moodle je potřebné vymyslet a dodržovat jednotnou konvenci v názvech a identifikátorech. Podle pravidel Moodlu není možné, aby identifikátor obsahoval čísla, velký písmena a speciální znaky. Pouze řetězec malých písmen je správný.

Pojmenování jsem zvolil identifikátor **linprog**, jako zkratku z názvu Lineární programování. Zmínění pojem úmyslně vybral, kvůli širokého množství využití a rozvoje pluginu. Oproti úzkému směru pouze se Simplexovým algoritmem.

3.3.3.2 Možnost realizace

Budu vycházet z programování celého modulu za použití prázdné šablony. Na základě uvedených možnostech v předchozích kapitolách, včetně jejich analýzách a východiska použití šablony. Z důvodu větší kontability budu plugin vyvíjet na jedné z nejnovějších verzí, abych byl modul déle udržitelný.

Jako nejvhodnější řešení se nabízí celý plugin naprogramovat od začátku. Než upravovat a provádět implementaci z modulu ze standardní instalace Moodlu. Tím také vznikne možnost celý plugin dále rozvíjet.

Datové struktuře se budu podrobněji věnovat v následující části.

V další kapitole se budu zabývat detailní implementaci, podle jednotlivých souborů v Moodlu.

3.3.3.3 Databáze

Pro budoucí plugin je potřeba navrhnout a vytvořit databázové tabulky. Nejprve musím navrhnout rozsáhlejší databázi, neboť jsem zvolil širší pojmenování modulu Lineární programování. A to s příslušnými hodnotami pro specifický výpočet podle požadavků.

Analýza databáze vytvářím plugin pro testovou otázku. Musím tedy zanalyzovat možnosti napojení či použití databázové struktury. Z uvedeného důvodu a organizace Moodlu je nutné se věnovat především tabulkám se zaměřením na *question*. Informace získané o databázovém schématu pochází přímo z dokumentace o Moodle databázi [22].

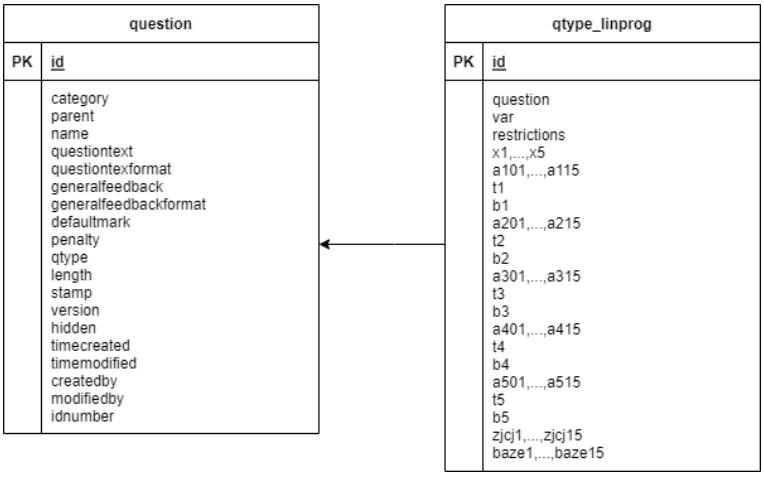
Na obrázku 3.4 Ukázka databázové tabulky Question je vidět část propojení s ostatními databázovými tabulky. Hned v úvodu tabulky u políčka *id* je vidět, že tato tabulka je napojená na testové otázky, na sebe sama, odpovědi a různá nastavení. Zmíněné další tabulky dědí z této hlavní tabulky a v systému Moodlu to znamená, že vše zmíněné je součástí otázky (question). Odkazuje na svou rodičovskou tabulku *question_categories*. Tabulka obsahuje celkem 20 položek. V rámci mého vývoje bude nutné napojit mnou vytvořenou tabulku na probíranou tabulku *question*. Díky modularitě Moodlu, bude zapotřebí správně adresářově zařadit můj plugin.

Návrh databáze databázová tabulka bude obsahovat, kromě povinných položek, místa pro uložení hodnot z formuláře pro úpravu testové úlohy: počet proměnných, počet podmínek, jednotlivé proměnné, atd. Z důvodu většího plánovaného použití pluginu navrhuji raději tabulku s více hodnotami. Například: větší počet proměnných a podmínek, dává možnost se dotazovat na cenové proměnné či na bázi.

Při návrhu databáze se nesmí zapomenout na požadovanou strukturu pojmenování tabulky obsahuje modulové zařazení v systému *qtype*. V poslední řadě tabulka obsahuje identifikátor, v mém případě
linprog. Spojením zmíněných názvů vznikne pojmenování databáze v systému **qtype_linprog**.

Pro lepší znázornění přikládám zjednodušenou databázovou strukturu v obrázku 3.5 Návrh umístění v databázové struktury. Novou tabulku pro můj plugin je nutné napojit na *question* tabulku, která bude rodičovská tabulka, jak je vidět na zmíněném obrázku. Tímto přidáním vznikne požadovaná struktura.

Obrázek 3.4: Ukázka databázové tabulky Question
Zdroj: snímek obrazovky Bc. Denise Drdy z Moodle LMS 3.9 Database



Obrázek 3.5: Návrh umístění v databázové struktury
Zdroj: vlastní zpracování Bc. Denise Drdy

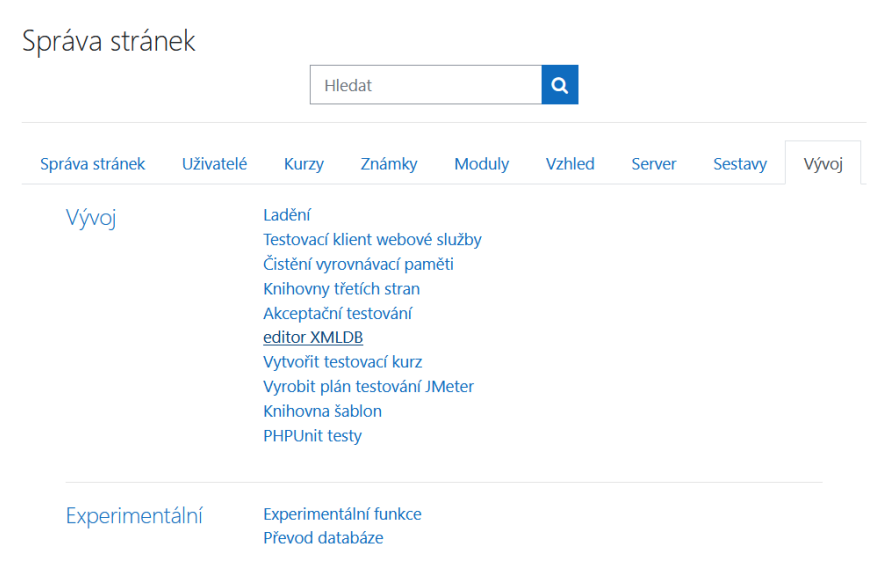
Principy fungování Do databázové tabulky se ukládají hodnoty sloužící pro výpočet příkladu z formuláře pro úpravu testové úlohy. Konkrétně se pracuje s adresářem *db* ve složce pluginu. Struktura databáze je uložena v souboru *install.xml*.

Existují dvě možnosti vytvoření tabulky do Moodlu. První je naprogramování skritpového kódu přímo ve zmíněném souboru *install.xml*. Tento způsob Moodle nedoporučuje [17], neboť lze se snadno odchýlit od požadované struktury Moodlu. Druhá možnost je vytvoření a úprava pomocí XMLDB editoru přímo v Moodlu. *„XMLDB editor je nástroj pro vytváření .xml souborů, které specifikují, jak má Moodle nastavit své databázové tabulky. Dříve museli vývojáři vytvářet samostatné instalační soubory .sql pro mysql a postgres, ale nyní je potřeba pouze databázově neutrální soubor, který podporuje mnohem více databází.“*
[17, Hlavní strana ▶ DB ▶ XMLDB]

Vytvoření databáze pro vytvoření databázové tabulky jsem se rozhodl použít doporučovaný postup přes XMLDB editor. Nejprve musím v adresáři mého pluginu vytvořit složku *db*, aby se vůbec v systému Moodlu zobrazila. V následujících podkapitolách se budu věnovat postupné práci s editorem.

Umístění editor se nachází v nastavení Moodlu, konrétně *Správa webu* ▶ *Vývoj* ▶ *XMLDB editor*. Pro lepší znázornění přikládám obrázek 3.6 Umístění XMLDB editoru.

Vyhledání následně je za zapotřebí najít hledanou databázi podle umístění v adresáři Moodlu. Jelikož programujeme testovou otázku, budu hledat v *question/type*. Poté stačí použít identifikátor, v mém případě *linprog*. Jediné místo kde se ukládá databáze je *db*. Spojením vznikne hledané umístění **question/type/linprog/db**. Databázi vytvořím pomocí tlačítka *vytvořit* a tím vznikne prázdný soubor *install.xml*.

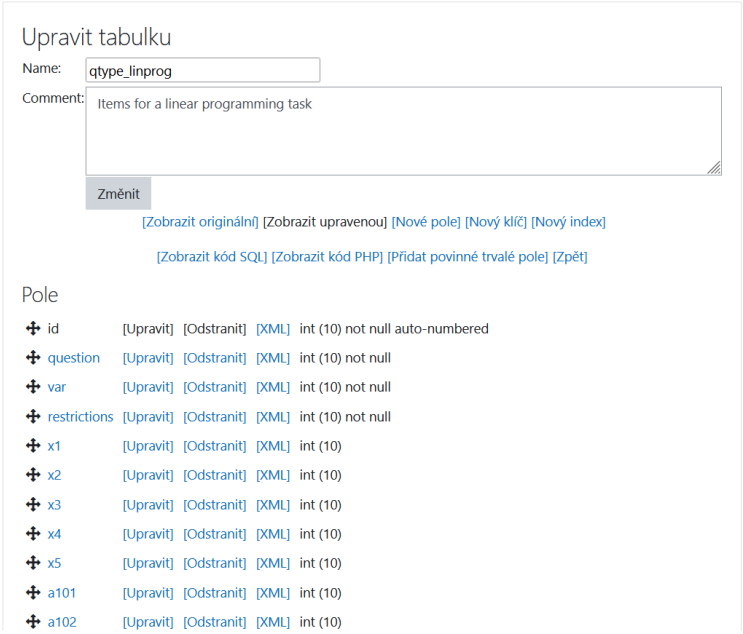


Obrázek 3.6: Umístění ve XMLDB editoru
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

Upravení XML souboru po otevření vyhledané databáze se zobrazí podrobnosti o XML souboru, jak je vidět na obrázku 3.7 Upravení XML souboru v editoru. Nalezneme zde umístění v systému, verzi systému a komentář k databázi. Dále máme mnoho možností od úpravy, přidávání tabulek, až po zobrazování kódu. V poslední části, lze vidět vytvořené tabulky.

Obrázek 3.7: Upravení XML souboru v editoru
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

Ukázka tabulky kromě povinných položek jsem doplnil tabulku o další položky podle předchozího návrhu. Na obrázku 3.8 Ukázky databázové tabulky v editoru je vidět název tabulky, která musí dodržet požadovaná pravidla Moodlu a komentář s popisem tabulky. Dále pak možnosti zobrazení, upravování, přidávání nových políček i indexů a v poslední řadě nesmí chybět ukládání tabulky. Následuje odíl Pole, kde jsou jednotlivé položky i s nabídkou různé úpravy a vlastnostmi.



Obrázek 3.8: Ukázka databázové tabulky v editoru
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

Aplikace do pluginu po vytvoření databáze podle předchozích rad. Nesmím zapomenout uložit tabulku do systému a tím se databáze uloží z vývojového prostředí do zdrojového kódu. Pro následnou aktualizaci celé databáze Moodlu je potřeba vytvořit soubor *upgrade.php* v příslušné databázové složce pluginu. Podle návodu ve vývojové dokumentace Moodlu [17].

3.3.4 Vývoj prototypu

Celý vývoj se skládá z několika kroků. Jedná se o programování modulových souborů v rámci pluginu. Každý soubor má svojí definici a různé vlastnosti, které se musí dodržet při vytváření přídavného modulu pro Moodle.

Postupně projdu jednotlivé akce, které je zapotřebí naprogramovat. Od definování formuláře, přes zajištění správné logiky na pozadí pluginu po zobrazování testu studentovi. Nesmí zde chybět, ani jazyková lokalizace.

Jelikož se jedná o vývoj prototypu pouze na Simplexový algoritmus. Po vytvoření nebudu nahrát plugin na schválení komunitou Moodlu. Tuto akci provedu, až po naprogramování všech testových úloh z lineárního programování.

3.3.4.1 Definování formuláře pro úpravu úlohy - edit_linprog_form.php

Pro dodefinování dalších formulářových políček pro úpravu úlohy slouží soubor *edit_linprog_form.php*, kde lze rozšířit základní formulář pro editaci úlohy. Jedná se o třídu **qtype_linprog_edit_form**, která dědí vlastnosti z **question_edit_form**. Tím, lze dodefinovat další políčka do testové otázky.

Základní formulář obsahuje následující políčka:

- Kategorie
- Název úlohy
- Text úlohy
- Výchozí známka
- Obecná reakce
- ID identifikační číslo

Všechny políčka formuláře ponechám a navíc doimplementuji další, podle potřeb Simplexového algoritmu. Dále ve zmíněném souboru naprogramuji další funkce: načtení dat do formuláře, včetně ošetřený chybně zadaných vstupných dat. V následnicích podkapitolách se budu věnovat jednotlivým metodám.

Definování dalších formulářových políček v závislosti na Simplexovém algoritmu dodefinuji následující políčka:

- Počet proměnných (1 až 5)
- Počet podmínek (1 až 5)
- x1,...,x5
- a11,...,a55
- Typ podmínky ke každé podmínce
- Pravá strana podmínky ke každé podmínce
- Odpověď

Formuláře v Moodlu používají syntaxy PEARu [23]. *„PEAR je framework a distribuční systém pro opakovaně použitelné komponenty PHP.“* [23, Hlavní stránka] Políčka ve formuláři se definují metodou *$mform->addElement*. Lze nastavit datový typ *$mform-> setType* a počátečné parametry *$mform-> setDefault*. V poslední řadě označení povinných políček *$mform->addRule*.

Celá definice formulářových políček probíhá ve funkci **definition_inner($mform)** Všechna zmíněné implementovaná políčka musí být povinná, kromě posledního s odpovědí. Poslední políčko se bude vypočítávat automaticky.

Listing 3.1: Ukázka kódu na dodefinování formulářových políček

```
function definition_inner($mform) {
    global $SESSION, $CFG, $DB;

    $mform = $this->_form; // Don't forget the underscore!

    //Number of variables
    $mform->addElement('text', 'var', get_string
        ('countvariable', 'qtype_linprog'), ['size' => 0]);
    $mform->setDefault('var', 3);
    $mform->setType('var', PARAM_INT);
    $mform->addRule('var', get_string
        ('requiredfield', 'qtype_linprog'), 'required');

    //Number of restrictions
    $mform->addElement('text', 'restrictions', get_string
        ('countrestriction', 'qtype_linprog'), ['size' => 0]);
    $mform->setDefault('restrictions', 3);
    $mform->setType('restrictions', PARAM_INT);
    $mform->addRule('restrictions', get_string
        ('requiredfield', 'qtype_linprog'), 'required');

    $vars = max(0, min(5, $this->question->options->var ?? 3));
```

```
    $restrictions = max(0, min(5,
        $this->question->options->restrictions ?? 3));
    $typecondition = ['<=', '>=', '='];

    for ($v~= 1; $v~<= $vars; ++$v) {
        $mform->addElement('text', 'x' . $v, get_string
            ('mx' . $v, 'qtype_linprog'), ['size' => 2]);
        $mform->setType('x' . $v, PARAM_INT);
        $mform->addRule('x' . $v, get_string
            ('requiredfield', 'qtype_linprog'), 'required');
    }

    //Display of conditions
    $mform->addElement('static', 'conditions', get_string
        ('conditions', 'qtype_linprog'));

    for ($r = 1; $r <= $restrictions; ++$r) {
        $mform->addElement('static', 'cond' . $r, get_string
            ('cond'.$r, 'qtype_linprog'));

        for ($v~= 1; $v~<= $vars; ++$v) {
            $name = sprintf('a%d%02d', $r, $v);
            $mform->addElement('text', $name, get_string
                ($name, 'qtype_linprog'), ['size' => 2]);
            $mform->setType($name, PARAM_INT);
            $mform->addRule($name, get_string
                ('requiredfield', 'qtype_linprog'), 'required');
        }

        //Select type of condition
        $mform->addElement('select', 't' . $r, get_string
        ('settypeofcondition', 'qtype_linprog'), $typecondition);

        //Variable b
        $mform->addElement('text', 'b' . $r, get_string
            ('b' . $r, 'qtype_linprog'), ['size' => 2]);
        $mform->setType('b' . $r, PARAM_INT);
        $mform->addRule('b' . $r, get_string
            ('requiredfield', 'qtype_linprog'), 'required');
    }

    //Answer
    $mform->addElement('static', 'computed_answer', get_string
        ('answer'));

    // Output the form
    if ($mform->validate()) {
        $mform->freeze();
    }
}
```

Načtení dat do formuláře načtení dat se provede pomocí podmínky načtení prázdných políček, pouze u nově definovaných formulářů. U základních políček stačí pouze volat rodičovskou funkci. Načtení hodnot se ukáže následujícím kódu.

Listing 3.2: Ukázka kódu na načtení hodnot do formuláře

```
public function set_data($question) {
    //load data to form
    $this->data_preprocessing($question);

    if (!empty($question->options) &&
        !empty($question->options->answers)) {
        $question->computed_answer = reset
            ($question->options->answers)->answer;
    }

    parent::set_data($question);
}
```

Ošetřený chybně zadaných vstupných dat u formulářových polí je nutné ošetřit vstupní data. Počet proměnných a podmínek nesmí být menší než nula a zároveň větší než pět. V následujícím kódu provedu ošetření chyb.

Listing 3.3: Ukázka kódu na ošetření chybně zadaných vstupných dat

```
public function validation($data, $files) {
    $errors = array();

    //fix count of variables less of the zero
    //and greater than five
    $var = $data['var'];
    if ($var <= 0) {
        $errors['var'] = get_string('varerrnull','qtype_linprog');
    } elseif ($var > 5) {
        $errors['var'] = get_string('varerr','qtype_linprog');
    }

    //fix count of restrictions less of the zero
    //and greater than five
    if ($data['restrictions'] <= 0) {
        $errors['restrictions'] = get_string
            ('restrictionserrnull', 'qtype_linprog');
    } elseif ($data['restrictions'] > 5) {
        $errors['restrictions'] = get_string
            ('restrictionserr', 'qtype_linprog');
    }

    if ($errors) {
        return $errors;
    } else {
```

```
        return true;
    }
}
```

3.3.4.2 Metody vyhodnocování pluginu - question.php

V této metodě naprogramuji automatické vyhodnocení úlohy a nastavení možností odpovědí pro studenty. Na základě předchozí analýzy jsem vybral jako vhodnou možnou optimální odpověď desetinné číslo a zaškrtávací políčko pro neexistující řešení. Dále je možné programovat různé chování odpovědi, či ošetřit nesprávný výsledek nebo prázdnou odpověď. V následujících podkapitolách se budu věnovat jednotlivým vybraným funkcím.

Shrnutí odpovědi vytvoření možných odpovědí, které vedou k danému shrnutí odpovědí. V následujícím kódu provádím shrnutí odpovědi.

Listing 3.4: Ukázka kódu na shrnutí odpovědi

```
public function summarise_response(array $response) {
    if (!empty($response['answer_no_solution'])) {
        return get_string('no_solution', 'qtype_linprog');
    }
    else if (isset($response['answer_numerator'])) {
        return sprintf('%f', $response['answer_numerator']);
    } else {
        return null;
    }
}
```

Ošetření vložení prázdné odpovědi ohodnocení odpovědí podle vyplnitelnosti. Lze nastavit možnosti správně, částečně nebo špatně v závislosti na typu úlohy. Zvolil jsem ošetření buď správná odpověď při správně zadaném výsledku, jinak špatná odpověď. V následujícím kódu provádím ošetření prázdné odpovědi.

Listing 3.5: Ukázka kódu na ošetření vložení prázdné odpovědi

```
public function get_validation_error(array $response) {
    if ($this->is_gradable_response($response)) {
        return '';
    }
    return get_string('pleaseenterananswer', 'qtype_linprog');
}
```

Získání správné odpovědi zmíněná metoda slouží k získání správné odpovědi studenta. Ta se převede na požadovaný formát, který se následně zobrazí ve formuláři. Je možné i vypočítanou odpověď převést na smysluplnější. V následujícím kódu provádím získání správné odpovědi.

Listing 3.6: Ukázka kódu na získání správné odpovědi

```
public function get_correct_response() {
    $correct = $this->qtype->calculate_answer($this->id);

    if ($correct === null) {
        return ['answer_no_solution' => true];
    } else {
        return ['answer_numerator' => $correct];
    }
}
```

Automatické vyhodnocení odpovědi jedná se o naprogramování logiky chování při vyhodnocení odpovědi. Musí se brát v potaz všechny skutečnosti v testu. Následně se vyhodnotí konečná známka pro aktuální pokus podle vypočítané odpovědi. V následujícím kódu se provede automatické vyhodnocení odpovědi.

Listing 3.7: Ukázka kódu na automatické vyhodnocení kódu

```
public function grade_response(array $response) {
    $correct = $this->qtype->calculate_answer($this->id);

    if ($correct === null &&
        $response['answer_numerator'] == 0) {
        $fraction = empty($response['answer_no_solution'])
            ? 0 : 1;
    } else if (empty($response['answer_no_solution'])){
        $answer =  $response['answer_numerator'];
        $fraction = $answer === $correct ? 1 : 0;
    }

    return array($fraction,
        question_state::graded_state_for_fraction($fraction));
}
```

3.3.4.3 Metody pluginu - questiontype.php

Ve zmíněném souboru se provádí specifické chování na pozadí celého modulu. Jedná se o uložení dalších formulářových políček do databáze, pomocí funkce *extra_question_fields*. Dále o výpočet odpovědi Simplexového algoritmu, včetně pomocných funkcí. V neposlední řadě se jedná o uložení vypočítané odpovědi do databázové tabulky.

V následujících podkapitolách se budu věnovat jednotlivým vybraným funkcím.

Výpočet odpovědi nejprve provedu načtení hodnot z databáze. Následně si přípravním data na aplikaci externí knihovny. Zjistím optimální řešení celé úlohy. Popřípadě řešení Simplexového algoritmu, převedu výsledek do přehlednějšího desetinného místa. Výpočet nemá řešení. V následujícím kódu provádím výpočet odpovědi.

Listing 3.8: Ukázka kódu na výpočet odpovědi

```
public function calculate_answer($questionId): ?float {
    global $DB;
    $params = $DB->get_record('qtype_linprog',
        ['question' => $questionId], '*', MUST_EXIST);
    $vars = [];

    for ($i = 1; $i <= $params->var; ++$i) {
        $vars['x' . $i] = $params->{"x$i"};
    }

    $task = new Simplex\Task(new Simplex\Func($vars));

    for ($i = 1; $i <= 5; ++$i) {
        $t = $params->{"t$i"};
        $b = $params->{"b$i"};

        if ($t === null || $b === null) {
            continue;
        }

        $vars = [];

        for ($j = 1; $j <= $params->var; ++$j) {
            $vars['x' . $j] = $params->
                {sprintf('a%d%02d', $i, $j)};
        }

        switch ($params->{"t$i"}) {
            case 0: $cond =
                Simplex\Restriction::TYPE_LOE; break;
            case 1: $cond =
                Simplex\Restriction::TYPE_GOE; break;
            case 2: $cond =
```

```
                Simplex\Restriction::TYPE_EQ; break;
        }

        $task->addRestriction
            (new Simplex\Restriction($vars, $cond, $b));
    }

    $solver = new Simplex\Solver($task);
    /** @var Simplex\Table[] $steps */
    $steps = $solver->getSteps();
    $last = end($steps);

    if ($last->hasSolution()) {
        $solution = $this->convert_to_decimal
            ($last->getZ()->getB()->toString());
        return $solution;
    } else {
        return null;
    }
}
```

Uložení vypočtené odpovědi jedná se následující uložení dat specifického nastavení vypočtené odpovědi. Pomocí funkce *save_question,* která je součástí rodičovské funkce *save_question_question*. Zde se rozlišuji odpověď s řešením jež nemá řešení. V následujícím kódu provádím uložení vypočtené odpovědi.

Listing 3.9: Ukázka kódu na uložení vypočtené odpovědi

```
public function save_question_options($question)
{
    parent::save_question_options($question);

    $answer = $this->calculate_answer($question->id);

    $question->answer = [
        $answer === null ? get_string
            ('nosolution', 'qtype_linprog') : $answer,
    ];

    $question->fraction = [1.0];
    $question->feedback = [['text' => get_string
        ('beautiful_solution_my_friend', 'qtype_linprog'),
        'format' => FORMAT_PLAIN]];
    $this->save_question_answers($question);
}
```

3.3.4.4 Externí knihovna na výpočet Simplexového algoritmu

Jak jsem již v jedné z předchozí kapitol uváděl jedná se o efektivnější využít existující externí knihovna na výpočet, které je jednoduší než vytváření vlastní knihovnu. Využil jsem tedy knihovnu Petra Kesslera[20]. Je to volně dostupná, využívaná licencí MIT (věnoval jsem ve své bakalářské práci [9]). V následujících částech se budu věnovat tomu nejdůležitějšímu v rámci citování licenčním podmínkách.

Licenční podmínky *„Hlavní výhodou je že šíření odvozených děl či výtvorů uživatele neomezuje. To znamená, že lze v odvozených dílech použít téměř vše bez porušení licenčních podmínek.“* [9, Strana 61] Z toho důvodu jsem se rozhodl licenční podmínky uvedené ve správci knihovny zveřejnit i ve své diplomové práci.

Listing 3.10: Správce knihovny Simplexového kalkulátoru [20]

```
{
    "name": "uestla/simplex-calculator",
    "type": "library",
    "description": "Simple tool for linear programming problems
       solving",
    "keywords": ["simplex", "algorithm", "linear", "programming"],
    "homepage": "http://simplex.tode.cz",
    "license": "MIT",
    "support": {
        "issues": "https://github.com/uestla/Simplex-Calculator/
           issues"
    },
    "require": {
        "php": ">=5.3.0"
    },
    "require-dev": {
        "nette/tester": "@dev"
    },
    "autoload": {
        "psr-0": {
            "Simplex": "/"
        }
    }
}
```

3.3.4.5 Zobrazení testu - renderer.php

Pro zobrazení dalších hodnot, v testu zadaném studentovi, slouží soubor *renderer.php*. Na základní zobrazení postačí čistý jazyk PHP, bez nutnosti JavaScriptu, který se používá k dynamickému zobrazení. Jedná se o definování co a jak se zobrazí studentovi při testu. Lze zobrazit zadané texty úloh i hodnoty v příkladě.

K základním zobrazovaným prvků přidám zadané hodnoty maximalizace a jednotlivé omezující podmínky. Dále umožním studentovi zapsat číselnou optimální odpověď do textového pole. V případě neexistujícího řešení vytvořím variantu pro zaškrtnutí políčka *nemá řešení*. V následujících podkapitolách se budu věnovat jednotlivým vybraným funkcím.

Vygenerování zobrazení zadaného testu zmíněná metoda slouží k zobrazení zadané části otázky testu. Jedná se o text úlohy, možnost pro zadání testové odpovědi. Doprogramuji zobrazení základní úlohy, včetně proměnných za omezujících podmínek a závěrem vytvořím kromě textového pole na vepsání odpovědi, také zaškrtávací políčko na odpověď bez řešení.

V následujícím kódu vytvářím celou logiku zobrazování testů studentům.

Listing 3.11: Ukázka kódu na vygenerování zobrazení zadaného testu

```
public function formulation_and_controls(question_attempt $qa,
                question_display_options $options) {
    global $DB;
    global $CFG;

    $question = $qa->get_question();

    $questiontext = $question->format_questiontext($qa);
    $params = $DB->get_record('qtype_linprog', ['question'
        => $question->id], '*', MUST_EXIST);
    $section = get_string('max', 'qtype_linprog');

    for ($i = 1; $i <= $params->var; ++$i) {
        $x = $params->{"x$i"};
        $section .= $this->print_expression_member($x, $i, $i === 1);
    }

    $section .= '<br />';
    $section .= get_string('conditions', 'qtype_linprog');
    $section .= '<br />';

    for ($i = 1; $i <= $params->restrictions; ++$i) {
        for ($j = 1; $j <= $params->var; ++$j) {
            $a~= $params->{sprintf('a%d%02d', $i, $j)};
            $section .= $this->print_expression_member
                ($a, $j, $j === 1);
        }
        switch ($params->{"t$i"}) {
            case 0: $section .= ' <= '; break;
```

```
            case 1: $section .= ' >= '; break;
            case 2: $section .= ' = '; break;
        }
        $section .= $params->{"b$i"};
        $section .= '<br />';
    }

    $result = html_writer::tag('div', $questiontext, array
        ('class' => 'qtext'));
    $result .= html_writer::tag('div', $section, ['class' => 'qtext']);

    $numerator = html_writer::empty_tag('input', [
        'type' => 'text',
        'name' => $qa->get_qt_field_name('answer_numerator'),
        'value' => $qa->get_last_qt_var('answer_numerator'),
        'id' => $qa->get_qt_field_name('answer_numerator'),
        'size' => 5,
        'class' => 'form-control d-inline',
    ]);

    $no_solution = html_writer::empty_tag('input', [
        'type' => 'checkbox',
        'name' => $qa->get_qt_field_name('answer_no_solution'),
        'checked' => $qa->get_last_qt_var('answer_no_solution'),
        'id' => $qa->get_qt_field_name('answer_no_solution'),
        'class' => 'form-check-input',
        'value' => '1',
    ]);

    $result .= html_writer::start_tag('div', array
        ('class' => 'ablock form-inline'));
    $result .= html_writer::tag(
        'label',
        get_string('answer', 'qtype_linprog'),
        ['for' => $qa->get_qt_field_name('answer_numerator')],
    );
    $result .= html_writer::tag('span', $numerator,
        ['class' => 'answer mx-2']);
    $result .= ' ';
    $result .= html_writer::tag(
        'label',
        $no_solution . ' ' . get_string('nosolution', 'qtype_linprog'),
        ['class' => 'form-check fitem', 'data-fieldtype' => 'checkbox'],
    );
    $result .= html_writer::end_tag('div');

    return $result;
}
```

3.3.4.6 Jazyková lokalizace

Jazykové lokalizační soubory jsou umístěny ve složce *lang*, která se nachází v příslušném pluginu. V podadresářích jsou umístěni konkrétní jazykové balíčky. Celý modul jsem se rozhodl rovněž vyvíjet v češtině a angličtině

Jedná se o zápis nových proměnných, které nebyli v jádru Moodlu definované. Jednotlivé proměnné lze zapsat pomocí tvaru:
$string[‘proměnná‘] = ‘text proměnné‘;

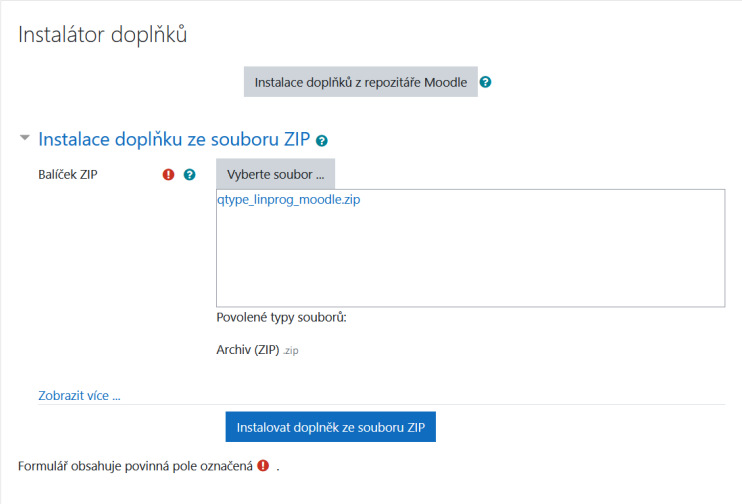
V následujících podkapitolách se budu věnovat jednotlivým překladům.

Česká lokalizace - cs/qtype_linprog.php skriptový soubor obsahující českou lokalizaci názvů, textů a řetězců pro vytvořený plugin.

Anglická lokalizace - en/qtype_linprog.php skriptový soubor obsahující anglickou lokalizaci názvů, textů a řetězců pro vytvořený plugin.

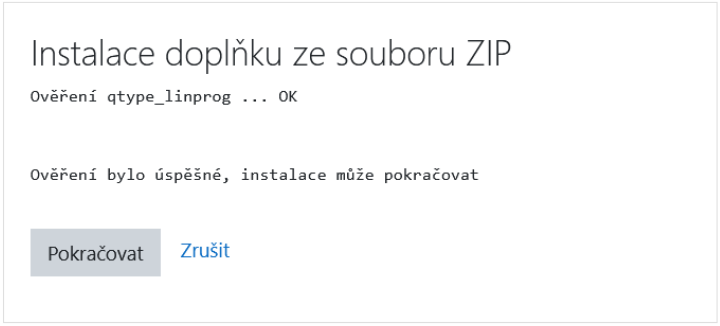
3.4 Instalace

Instalace pluginu provádí oprávněná osoba, většinou administrátor systému Moodlu. Celá instalace se řídí stejným procesem jako instalování zásuvných modulů, stažených z repositáře Moodle [24], ve formátu ZIP. Pro lepší znázornění přikládám obrázek 3.9 Instalace pluginu.



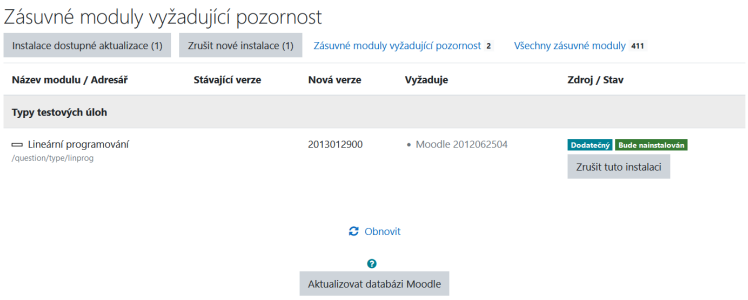
Obrázek 3.9: Instalace pluginu
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

Vytvořený plugin je přílohou mé diplomové práce. Nainstalování nového doplňku probíhá v příslušné sekci, konkrétně *Správa stránek* ▶ *Moduly* ▶ *Instalace doplňků*. Soubor **qtype_linprog_moodle.zip** vyberu a následně nahraji do systému Moodle.



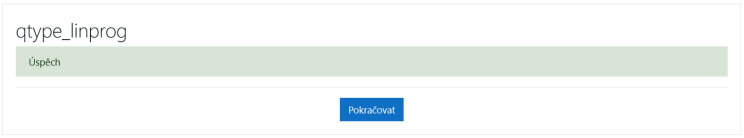
Obrázek 3.10: Ověření doplňku
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

Poté proběhne ověření instalace, jak lze vidět na obrázku 3.10 Ověření doplňku, s oznámením přímo u názvu pluginu. Dále se zobrazí hláška o úspěšné ověření s možností pokračování v instalaci.



Obrázek 3.11: Kontrola zásuvných modulů
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

Následuje aktualizace databáze Moodle. Pro lepší názornost přikládám obrázek 3.11 Kontrola zásuvných modulů, kde Moodle upozorňuje na zásuvné moduly, které vyžadují pozornost. V tomto případě se jedná o můj vytvořený plugin Lineárního programování.



Obrázek 3.12: Úspěšná instalace
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

Po provedení úspěšné aktualizace databáze Moodle se zobrazí oznámení o úspěchu. Jak lze vypozorovat z obrázku 3.12 Úspěšná instalace. Po dokončení instalace je plugin připraven k použití.

3.4.1 Podpora starších verzí

Vytvořený prototyp pluginu není možné nainstalovat na starší verze systému Moodle. Pro tento krok jsem se rozhodl pracovat z důvodu zbytečně neefektivní implementace na starých a v dnešní době skoro nepoužívaných verzích. Nemožná přenositelnost vychází z fungování rozhraných jednotlivých verzí Moodlu. Například dřívější verze řešila zobrazování testové otázky studentům pomocí jazyku HTML a v aktuální verzi zmíněné zobrazení se řeší vykreslení pomocí PHP, popř. dalších programovacích jazyků. Tímto vznikl plnohodnotný prototyp pluginu pro testování Simplexového algoritmu, fungující na aktuální verzi Moodlu.

3.5 Uživatelské rozhraní

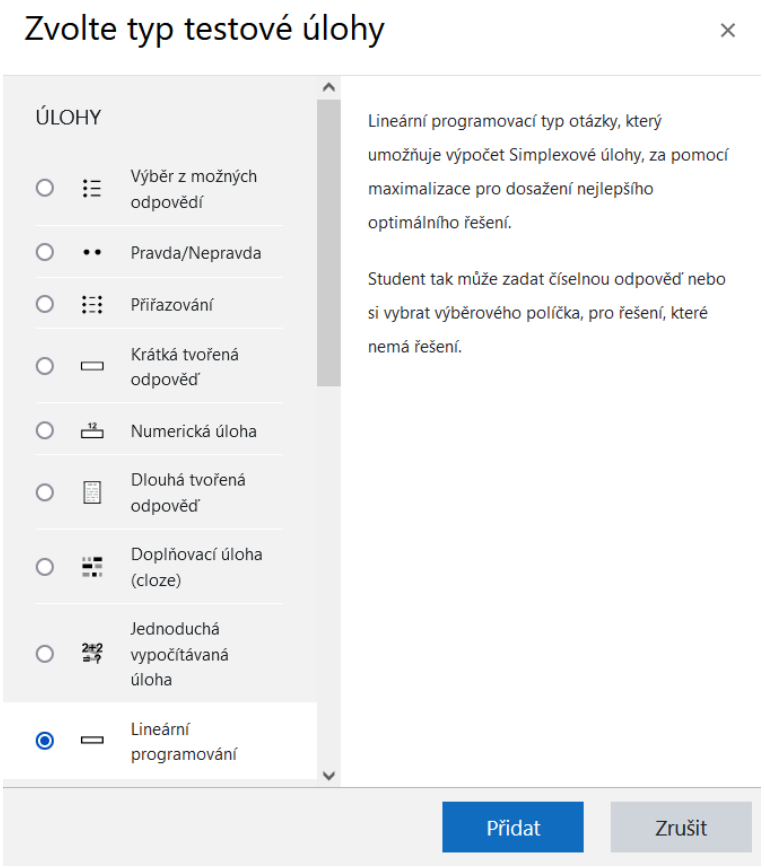
Uživatelské rozhraní prototypu pluginu se dělí na dvě skupiny uživatelů. Především se rozděluje podle jejich oprávnění. Jedná se role učitele či správce kurzu a role studenta. Každá ze zmíněných skupin má odlišná práva, různé možnosti ovládání.

Následující podkapitoly se budou zabývat výše uvedeným skupin v sytému Moodle, včetně jeho návodu k použití. Poté se testová úloha vloží obvyklým způsobem do konkrétního testu.

3.5.1 Rozhraní pro učitele

V rámci role učitele mám možnost přidávat nové testové otázky. Po nainstalováním pluginu, většinou správcem kurzu, je běžně dostupný všem jako ostatní úlohy.

Přidání nového zadání se provede při přidání testové úlohy do testu nebo do Banky úloh. Pro lepší znázornění přikládám obrázek 3.13 Výběr pluginu, kde je dostupný v seznamu mezi ostatními úlohami. Po nalezení a vyčlennění požadovaného modulu, stačí kliknout na tlačítko *Přidat*.

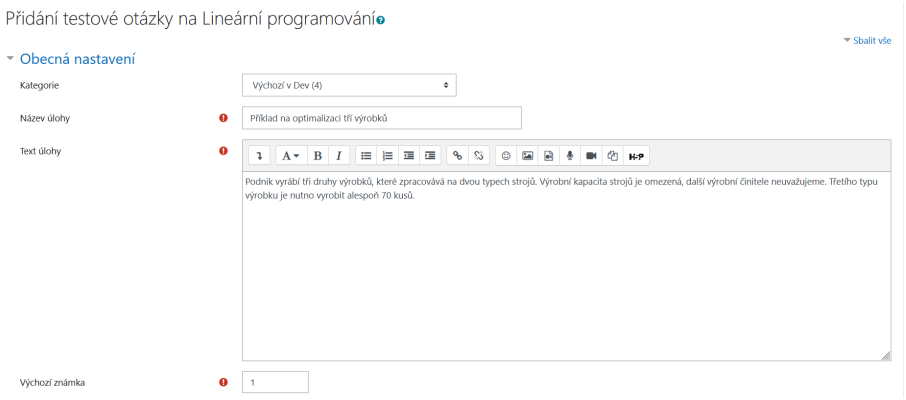


Obrázek 3.13: Výběr pluginu
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

Následně se zobrazí *Přidání testové otázky na Lineární programování*, kde se zobrazí formulář s množnostmi na zadání celého testu. Z důvodu lepšího vysvětlení fungování rozhraní pluginu, budu rovnou provádět názornou ukázku při zadávání příkladu na optimalizace u Simplexový algoritmu. Budu se věnovat úloze se třemi výrobky. V následujících podkapitolách rozdělím podle jednotlivých částí.

3.5.1.1 Základní nastavení

Základní nastavení zůstalo neměnné oproti ostatním úlohám. Ovládá se stejně jako v ostatních modulech. Pro uvedený příklad přikládám obrázek 3.14 Základní nastavení testové otázky, ve které jsem vyplnil povinné políčka *Název úlohy* a *Text úlohy*.



Obrázek 3.14: Základní nastavení testové otázky
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

3.5.1.2 Nastavení počtu proměnných a podmínek

Ve zmíněné části se zadává počet proměnných a omezujících podmínek. Vzhledem k rozsahu hodnot od 1 do 5, je nastavená výchozí hodnota 3, viz obrázek 3.15 Nastavení počtu proměnných a podmínek. Tuto hodnotu lze různě měnit v závislosti na potřebném počtu. Při změně hodnoty a následném kliknutí tlačítka *Uložit změny a pokračovat v úpravách* se příslušná následující políčka aktualizují podle zadaných hodnot.

Obrázek 3.15: Nastavení počtu proměnných a podmínek
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

3.5.1.3 Zadání jednotlivých proměnných

V našem příkladě jsem zadal tři proměnné. Jsou vidět jako vyžadované hodnoty. Jak lze vypozorovat z obrázku 3.16 Zadání jednotlivých proměnných.

Obrázek 3.16: Zadání jednotlivých proměnných
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

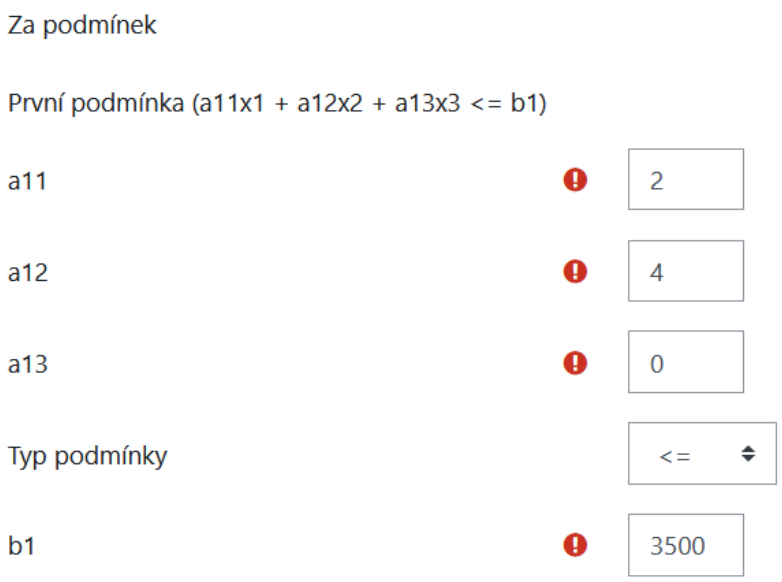
3.5.1.4 Zadání omezujících podmínek

Jelikož jsem zvolil pro příklad tři omezující podmínky, bude zapotřebí vyplnit všechny tři podmínky.

V následujících podčástech se budu věnovat každé jednotlivě.

První podmínka má tvar: ***2x1 + 4x2 <= 3500***. Zde následně vyplním příslušná políčka.

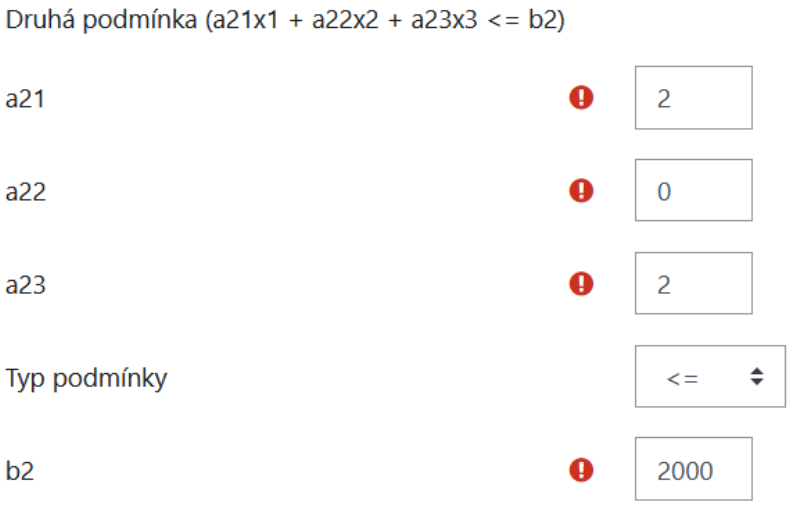
Pro lepší znázornění přikládám obrázek 3.17 Nastavení první podmínky.



Obrázek 3.17: Nastavení první podmínky
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

Druhá podmínka má tvar: ***2x1 + 2x3 <= 2000***. Zde následně vyplním příslušná políčka.

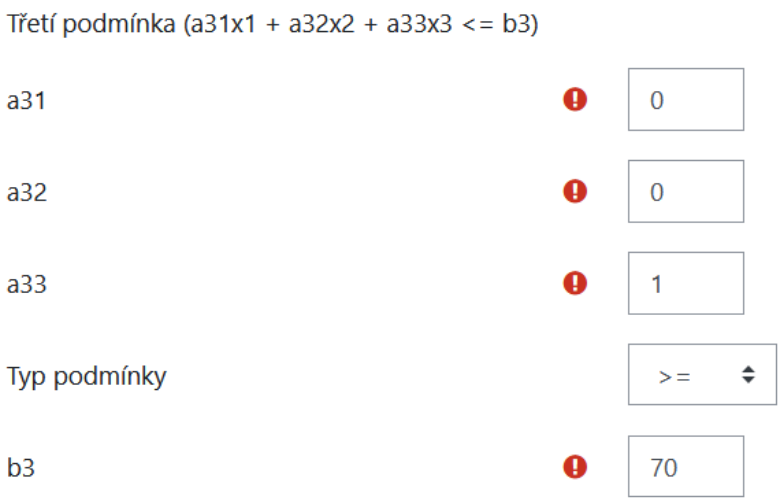
Pro lepší znázornění přikládám obrázek 3.18 Nastavení druhé podmínky.



Obrázek 3.18: Nastavení druhé podmínky
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

Třetí podmínka má tvar alespoň *70* kusů třetího výrobku.
Rovnicový tvar: *x3* ***>= 70***. Zde následně vyplním příslušná políčka.

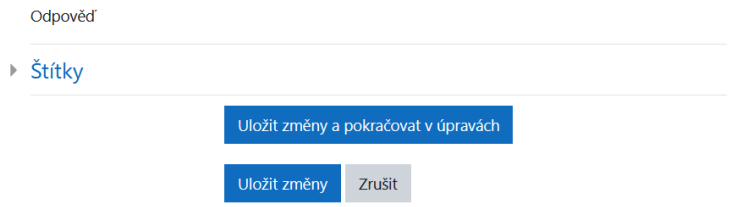
Pro lepší znázornění přikládám obrázek 3.19 Nastavení třetí podmínky.



Obrázek 3.19: Nastavení třetí podmínky
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

3.5.1.5 Výpočet úlohy

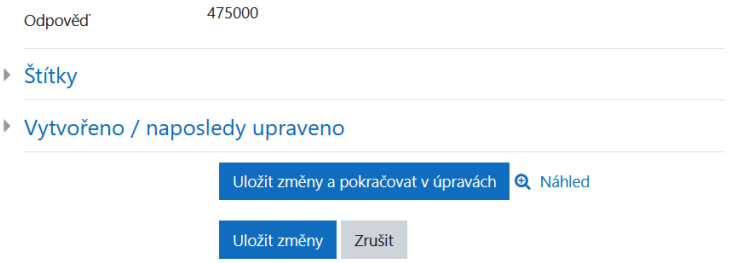
Po doplnění potřebných formulářových políček, lze vidět prázdné políčko *Odpověď*. (Jak lze vypozorovat z obrázku 3.20 Výpočet úlohy.) Pro získání správné odpovědi sloužící pro kontrolu, je nutné kliknout na tlačítko *Uložit změny a pokračovat v úpravách*



Obrázek 3.20: Výpočet úlohy
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

3.5.1.6 Odpověď úlohy

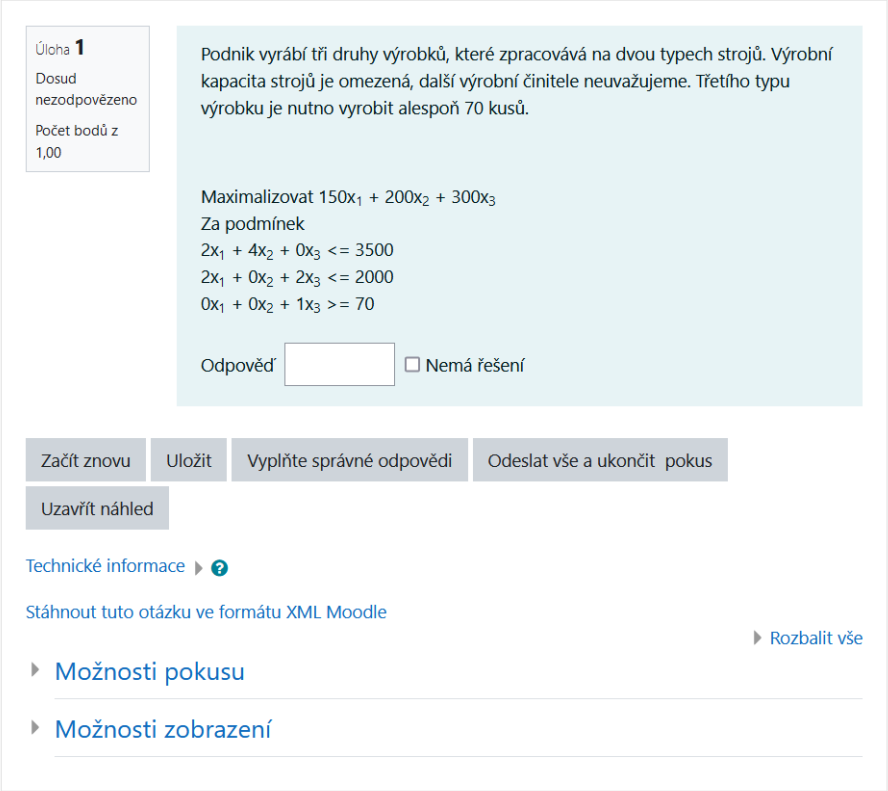
Po provedení předchozí akce se zobrazí správná odpověď. Zmíněná výsledná hodnota slouží i jako odpověď v ostrém testu. Pro lepší znázornění přikládám obrázek 3.21 Odpověď úlohy. Dokončením zmíněné předchozí akce, se zobrazí tlačítko *Náhled*, kterému se budu věnovat v následující části.



Obrázek 3.21: Odpověď úlohy
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

3.5.1.7 Náhled úlohy

Po rozkliknutí stejně nazvaného tlačítka se zobrazí náhled úlohy. Pro představu přikládám obrázek 3.22 Náhled úlohy. Kde se zobrazuje text úlohy a rovnice, které je potřeba maximalizovat za omezujících podmínek. Pod rovnicemi je sekce pro zápis odpovědi. Buď v číselném desetinném formátu, nebo zaškrtnutí políčka *Nemá řešení*.



Obrázek 3.22: Náhled úlohy
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

Poté následují tlačítka na ovládání úlohy, kde lze začít úlohu znovu, uložit, či vyplnit správné odpovědi. Až po odeslání pokusu a uzavření náhledu. V dolní části lze nalézt technické informace sloužící pro vývojáře. Možnost stáhnout vyplněnou šablonu ve formátu XML Moodle vhodná například na hromadný export a import úloh mezi systémy. Závěrem náhledu je možné nastavit možnosti pokusu a zobrazení úlohy.

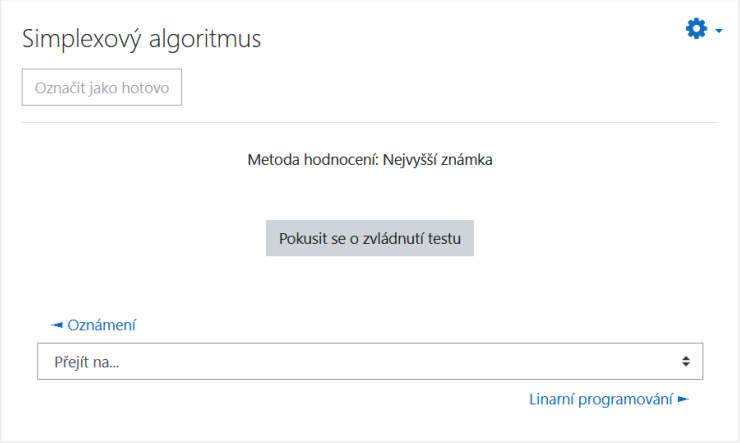
3.5.2 Rozhraní pro studenty

Po zpřístupnění testu, se studentovi zobrazí v nabídce Moodlu test, který je možné zobrazit, vyplnit a následně odevzdat.

V následujících podkapitolách se budu stručně věnovat studentskému ovládání.

3.5.2.1 Spuštění testu

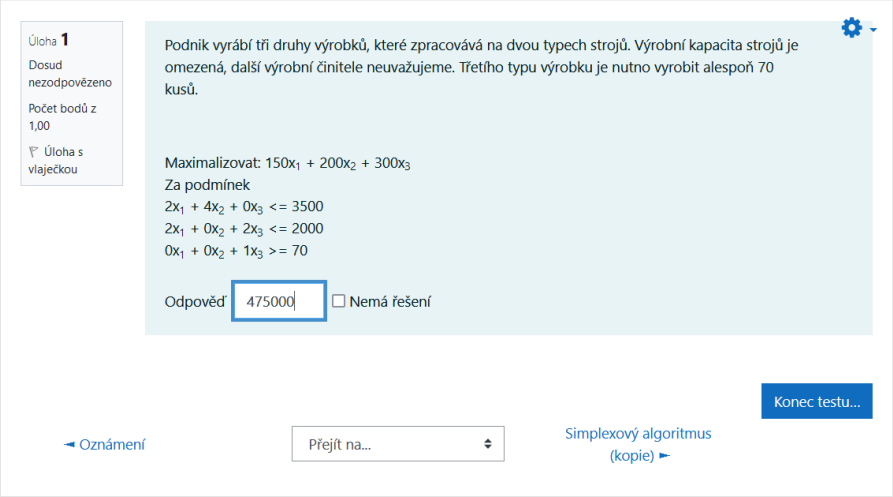
Student spustí test obvyklým způsobem, ten je neměnný. Pro lepší znázornění přikládám obrázek 3.23 Spuštění testu. Kde se student dozví metodu hodnocení. Pro pokračování musí kliknout na tlačítko *Pokusit se o zvládnutí testu*. Poté se zobrazí zadání testové úlohy.



Obrázek 3.23: Spuštění testu
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

3.5.2.2 Zadání testové úlohy

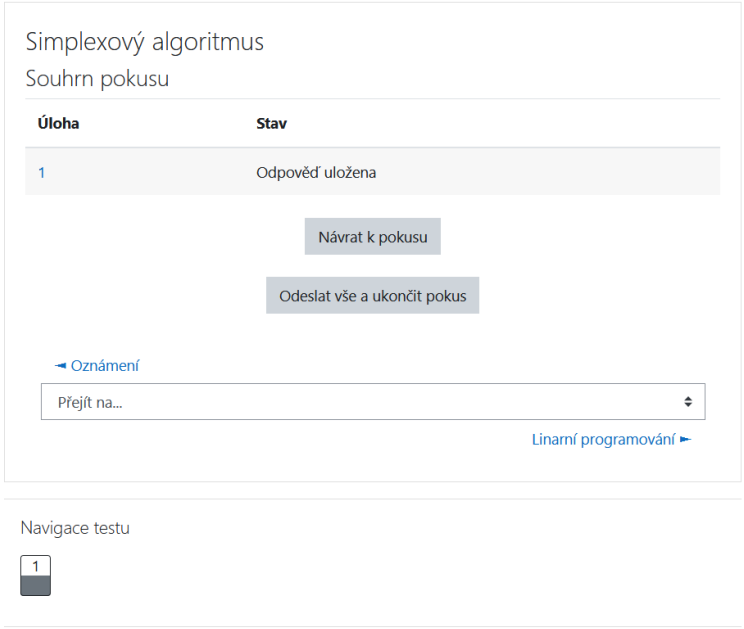
V záhlaví úlohy je číslo úlohy, status o zodpovězené otázky a další podrobnosti. Student vidí textové zadání úlohy. Hodnoty, které je potřeba maximalizovat a všechny omezující podmínky. V dolní části u odpovědi je políčko pro vložení číselné odpovědi se zaškrtávací možnost pro příklad bez řešení. Na obrázku 3.24 Zadání testové úlohy, je vidět zadání správné odpovědi. Po vyplnění odpovědi je zapotřebí kliknout na tlačítko *Konec testu*



Obrázek 3.24: Zadání testové úlohy
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

3.5.2.3 Souhrn pokusu

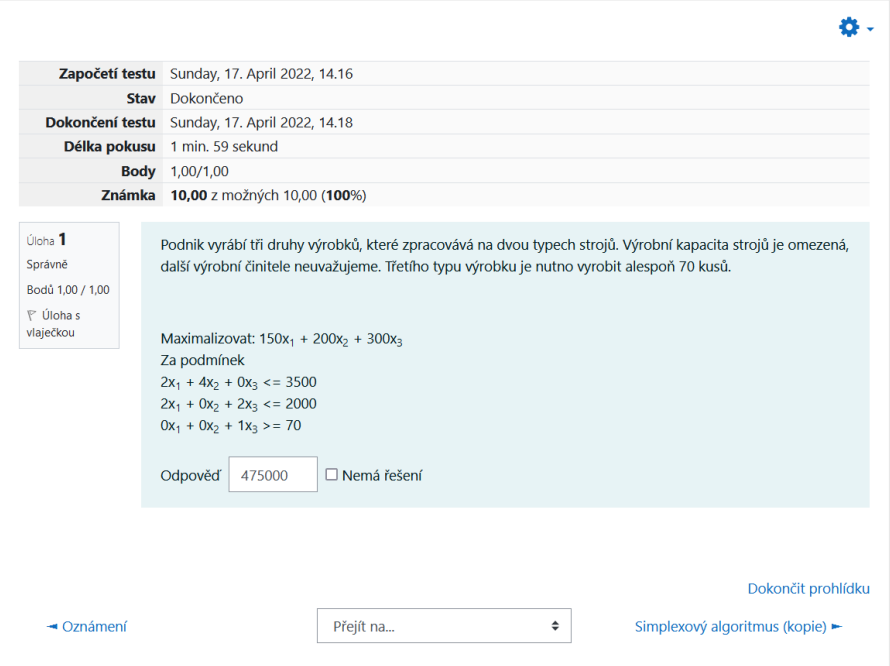
Na následujícím obrázku 3.25 Souhrn pokusu jsou informace o uložené odpovědi. S možností vrátit se zpět do úlohy nebo ukončit test. Po kliknutí na tlačítko *Odeslat vše a ukončit pokus* se zobrazí informace o tom, že po odevzdaní testu nelze měnit své odpovědi. Po odsouhlasením se zobrazí prohlídka testu s vyhodnocení výsledku.



Obrázek 3.25: Souhrn pokusu
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

3.5.2.4 Prohlídka testu s vyhodnocením

Po odevzdaní testu do systému Moodlu se zobrazí revizní prohlídka testu. Na této stránce lze nalézt informace o testu, stavu dokončení, bodového zisku a známky. Rovněž je zobrazena celá úloha jako u zadání testové úlohy, kde je zadám pouze výsledek ohodnocení. Pro lepší znázornění přikládám obrázek 3.26 Prohlídka testu s vyhodnocením.

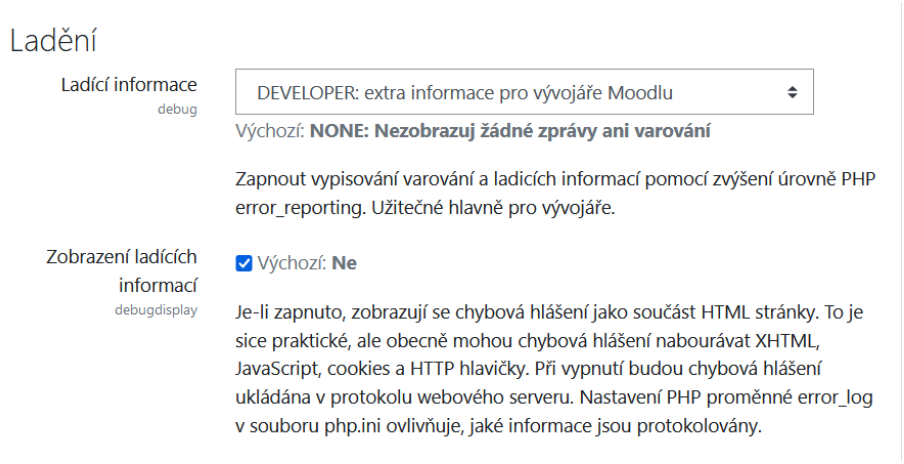


Obrázek 3.26: Prohlídka testu s vyhodnocením
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

3.6 Testování

Testování a ladění pluginu probíhalo již během jeho vývoje. Důvodem bylo odhalení případných logický či syntaktický chyb ve skriptech či nastavení. Programování probíhalo se zapnutým ladění pro vývojáře, které lze zapnout v nastavení *Správa stráněk* ▶ *Vývoj* ▶ *Ladění*.

Pro lepší znázornění přikládám obrázek B.5 Režim ladění, kde je vidět hodnota *DEVELOPER: extra informace pro vývojáře Moodlu* nastavená jako ladící informace. Toto nastavení zobrazovalo veškeré chyby, překlepy a varování v souborech PHP. Podle vývojářské dokumentace [17] se výrazně nedoporučuje používat zmíněný režim ladění. Z důvodů možných chybovým hláškám, které mohou být zneužitý nežádoucímy osoby.



Obrázek 3.27: Režim ladění
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

Při ověření a testování správné funkčnosti modulu je potřebné provést testování podle následujícího seznamu úkolů testů:

- Instalace/odinstalace pluginu
- Vytvoření různých typů otázek
- Kontrola a validace správnosti vyplnění editačního formuláře
- Výpočet správně odpovědi při zadání hodnot
- Zobrazení náhledu otázky
- Kontrola správnosti odpovědi
- Export otázky do podporovaného typu XML Moodle
- Úprava testové úlohy
- Vyplnění testu v roli studenta
- Kontrola možnosti kombinace správné textové odpovědi se zaškrtnutí políčka nemá řešení
- Okamžitý výsledek pro studenta
- Procházení pokusů a revize testů
- Česká lokalizace
- Anglická lokalizace

Při testování byli zjištěni chyby a překlepy v rámci jazykových lokací. Podle výsledků testů jsem závady odstranil a provedl opětovné testování podle zmíněného seznamu úkolů.

Změněný testovací scénář byl úspěšně otestován. Testování probíhalo na operačním systému Windows 10 na nainstalované verzi Moodle 3.11.4. v internetovém prohlížeči Firefox 99.0.1.

3.7 Dokumentace

Dokumentace zdrojového kódu se řídí podle pravidel pro přispívání a vývoji kódu Moodle[17]. To znamená, že PHP skripty jsou okomentovány podle požadavků Moodlu. Metody ve zmíněných souborech jsou převážně použity ze šablonových metod a ty jsou přizpůsobené podle požadavků na plugin. Ostatní metody se volají standardně bez úprav. Jelikož dokumentace je velmi užitečná pro další vývoj. Rozhodl jsem se využít software na tvorbu dokumentace Doxygen[25], který generuje dokumentaci ze zdrojových kódu, použitých metod atd. Vygenerovaná dokumentace ve formátu HTML je přílohou této práce.

Kapitola 4

Vyhodnocení a doporučení další postupu

V této části se budu postupně věnovat dvou hlavním věcem. První je zhodnocení vytvořeného prototypového pluginu. Druhá je doporučení dalšího postupu s návrhem zlepšení a rozvoje modulu.

4.1 Zhodnocení pluginu

Vytvořený prototyp pluginu má podle mého názoru velký potenciál ve využití pro tvorbu testů, neboť tento problém, testování matematických úloh z lineárního programování, zatím nikdo neřešil. Jedná se o prototypový modul, který umí na základě zadaných hodnot vypočítat optimalizaci pomocí Simplexového algoritmu. Zároveň jsem splnil, požadavky na modul. Učitel vidí ihned výsledek úlohy sloužící ke kontrole správného zadání. Lze díky rozhraní Moodlu celý test zduplikovat a upravit zadání jako jinou variantu. Následně lze zadat test studentům (ten může být odlišný). Navíc vytvořený modul umí provádět automatické vyhodnocení testu, které je vhodné nejen pro studenty, ale i pro učitele. Studenti okamžitě vidí svůj výsledek testu. Učitel nemá příliš velkou práci s hodnocením, pouze musí napsat a zadat zadání testu. Jelikož se jedná o prvotní prototyp modulu na tvorbu testu, nebyl nahrán, ani schválen komunitou Moodle. Rád bych ho dále efektivně rozvíjel o další funkcionality. Tomu se budu věnovat v následující podkapitole.

4.2 Doporučení dalšího postupu

Hlavní pozitivum rozšíření vidím, rozšířit o další funkcionality. Především o další matematické úlohy, konkrétně o: Dopravní úlohy, Duální modely, atd. Neboť jsem při návrhu plugin nazval: *Lineární programování* a v rámci implementovaného prototypu jsem řešil pouze optimalizační problém za pomocí Simplexového algoritmu.

Další rozvoj vytvořeného prototypu navrhuji rozdělit na celé testové otázky na části. Jednalo by se o možnost dotazovaní na postup výsledků a celých Simplexových tabulek. Současnou verzi, která hodnotí výslednou optimalizaci. Hodnocení úlohy podle jednoho výsledku se mi zdá poněkud nespravedlivé pro studenta. Neboť se může stát, že student bude při zkoušce ve stresu. Udělá překlep ve výsledku a systém vyhodnotí úlohu jako nesprávnou. Podle mého názoru je i důležitější postup a jakým způsobem se student dopracuje k výsledku.

Vytvořený prototyp lze snadno rozvíjet a rozšiřovat, než bude naimplementován dokonalý nástroj na testování složitých matematických úloh. Uvedl jsem pár návrhů na doporučení postupu. Z těchto uvedených důvodů je pro mě prioritou dokončený zmíněného rozšíření, předtím než nahraji modul mezi komunitu Moodlu. A stane se z mého pluginu oficiální zásuvní modul.

Závěr

Hlavnímu cíli práce, kterým bylo vytvoření prototypu na zadávání speciálních algoritmů v prostředí Moodlu, jsem se začal věnovat až po nastudování všech potřebných informací. Konkrétně jsem se věnoval zadefinování speciálního algoritmu., vybrání a důkladný popis s ukázkou výpočtu Simplexového algoritmu ze skupiny úloh Lineárního programování. Zmíněný algoritmus jsem se snažil vhodně využít pro implementaci prototypu.

Po definování a provedení jednoho ze základných požadavků, sběrů požadavků na zadávání speciálních algoritmů. Jsem se mohl naplno věnovat průzkumem možných řešení v rámci Moodlu. Stručně jsem analyzoval celý systém Moodle, od běžného používání, přes různé požadavky a informace po vývoj v tomto prostředí. Prozkoumal jsem možnou tvorbu nových pluginů a zároveň jsem zanalyzoval současné možnosti pro testové úlohy. Zaměřil jsem se na pravidla programování modulů, přispívání a sdílení naimplementovaných pluginů.

Dalším cílem bylo navrhnout prototypu na zadávání speciálních algoritmů v prostředí Moodlu. Nejprve jsem musel analyzovat možný vývoj, vymyslet účelné a výstižné pojmenování modulu. Součástí návrhu bylo vytvoření vhodnou strukturu databáze, která bude přijatelná pro další rozvoj prototypu. Následně jsem naimplementoval prototyp pluginu a v práci popsal s ukázky kódu. Vypracoval jsem uživatelský návod, kde se především zaměřuji na instalaci a uživatelské použití modulu, jak ze strany tvůrce testu, tak i ze strany studenta. Nechybí, ani jeden z posledních cílů věnující testování a dokumentaci pluginu. Testování probíhalo již během vývoje, ale vytvořil jsem i testovací scénář na ověření správné funkčnosti modulu. Závěrem této práce jsem zhodnotil celý naimplementovaný prototyp a doporučil další možný postup pro zásuvný modul na tvorbu testových otázek.

Celá práce byla pro mě výzvou. Zpočátku jsem neměl žádné zkušenosti s výpočtem Simplexového algoritmu, ani v programování v jazyce PHP a ani pro implementaci zásuvného modulu pod volně dostupným systémem. Vše jsem si musel nastudovat a následně procvičit. Z těchto důvodů mi práce přinesla obrovské zkušenosti ve zmíněných činnostech, které jsem zvládl i přes jejich časovou náročnost. Měl jsem možnost se zapojit do tvorba a rozvoje vzdělávacího systému. Díky velké podpoře svého okolí jsem se nevzdal a výzvu, podle mého

názoru, úspěšně dokončil.

Literatura

[1] Šubrt, T.; kol.: *Ekonomicko-matematické-metody*. Aleš Čeněk, s. r. o., třetí vydání, 2019, ISBN 978-80-7380-762-7.

[2] Brožová, H.; Houška, M.: *Základní metody systémové analýzy*. Česká zemědělská univerzita v Praze, první vydání, 22.10.2002, ISBN 80-213-0951-2.

[3] Volek, J.; Linda, B.: *Lineární programování*. Univerzita Pardubice, třetí vydání, listopad 2009, ISBN 978-80-7395-207-5.

[4] Dombek, D.; kol.: *Lineární algebra*. KAM FIT ČVUT v Praze, LS 2019/2020.

[5] Moodle Dokumentace. [online], [cit. 12.8.2021]. Dostupné z: `https://docs.moodle.org/archive/cs/Hlavn%C3%AD_strana`

[6] Moodle Documentation. [online], [cit. 12.8.2021], Vlastní překlad. Dostupné z: `https://docs.moodle.org`

[7] Váňová, T.; Váňová, A.: Moodle v síti. 2009.

[8] Drhlík, M.; kol.: *Moodle Kompletní průvodce tvorbou a správou elektronických kurzů*. Computer Press, první vydání, 2013, ISBN 978-80251-3759-8.

[9] Drda, D.: Věnná města českých královen - analýza projektu a rizik. 2020, bakalářská práce.

[10] Maněna, V.; kol .: *Moderně s Moodlem*. CZ.NIC, první vydání, 2015, ISBN 978-80-905802-7-5.

[11] Apache HTTP server project. [online], [cit. 27.10.2021]. Dostupné z: `https://httpd.apache.org/`

[12] MySQL. [online], [cit. 27.10.2021]. Dostupné z: `https://www.mysql.com/`

[13] PHP: Hypertext Preprocessor. [online], [cit. 27.10.2021]. Dostupné z: `https://www.php.net/`

[14] Buchtela, D.: PRŮVODCE SYSTÉMEM MOODLE ... pro pedagogy a GAELPy. [online], [cit. 7.10.2021]. Dostupné z: https://dl.webcore.czu.cz/file/WUVHZW9DZmdqZ0k9

[15] Vynikarová, D.: PRŮVODCE SYSTÉMEM MOODLE 2.5 pro GAELPy a pedagogy. [online], [cit. 7.10.2021]. Dostupné z: https://docplayer.cz/44419233-Pruvodce-systemem-moodle-2-5-pro-gaelpy-a-pedagogy-dana-vynikarova-sis-oikt-czu.html

[16] Moodle Demo - Mount Orange School. [online], [cit. 7.10.2021]. Dostupné z: https://school.moodledemo.net/my/

[17] Moodle Dokumentace pro vývojáře. [online], [cit. 1.11.2021], Vlastní překlad. Dostupné z: https://docs.moodle.org/dev/

[18] Moore, J.; Churchward, M.: *Moodle 1.9 Extension Development*. Publishing Ltd. 32 Lincoln Road Olton Birmingham, B27 6PA, UK, první vydání, duben 2010, ISBN 978-1-847194-24-4, vlastní překlad.

[19] Doporučení standardů PHP. [online], [cit. 7.11.2021], Vlastní překlad. Dostupné z: https://www.php-fig.org/psr/

[20] Simplex Calculator: PHP tool for linear programming problems solving. [online], [cit. 12.4.2021], Vlastní překlad. Dostupné z: https://github.com/uestla/Simplex-Calculator

[21] Question type template. [online], [cit. 11.4.2021], Vlastní překlad. Dostupné z: https://github.com/jamiepratt/moodle-qtype_TEMPLATE

[22] Moodle LMS 3.9 Database: Question Table Schema (question). [online], [cit. 13.4.2021], Vlastní překlad. Dostupné z: https://moodleschema.zoola.io/tables/question.html

[23] PEAR - PHP Extension and Application Repository. [online], [cit. 14.4.2021], Vlastní překlad. Dostupné z: https://pear.php.net/

[24] Moodle - plugins. [online], [cit. 15.4.2021], Vlastní překlad. Dostupné z: https://moodle.org/plugins/

[25] Doxygen - Generate documentation from source code. [online], [cit. 18.4.2021], Vlastní překlad. Dostupné z: https://www.doxygen.nl/index.html

Příloha **A**

Seznam použitých zkratek

GNU Svobodný operační systém
(anglicky Free operating system)

MOODLE Modulární objektově orientované výukové prostředí
(anglicky Modular Object-Oriented Dynamic Learning Environment)

LMS Systém řízení výuky
(anglicky Learning management system)

GUI Grafické uživatelské prostředí
(anglicky Graphical user interface)

XML Rozšiřitelný značkovací jazyk
(anglicky Extensible markup language)

HTML Hypertextový značkovací jazyk
(anglicky Hypertext Markup Language)

ZIP Soubor se zkomprimovanými daty
(anglicky A file with compressed data)

SCORM Referenční model objektu obsahu ke sdílení
(anglicky Shareable Content Object Reference Model)

URL Jednotný lokátor zdroje
(anglicky Uniform Resource Locator)

PDF Přenosný formát dokumentů
(anglicky Portable Document Format)

PNG Přenosná síťová grafika
(anglicky Portable Network Graphics)

DOC Dokument
(anglicky Document)

XLS Formát MS Excel
(anglicky MS Excel format)

GPL Obecná veřejná licence
(anglicky General Public License)

MIT Svobodná licence vzniklá na Massachusettském technologickém institutu
(anglicky Free license created at the Massachusetts Institute of Technology)

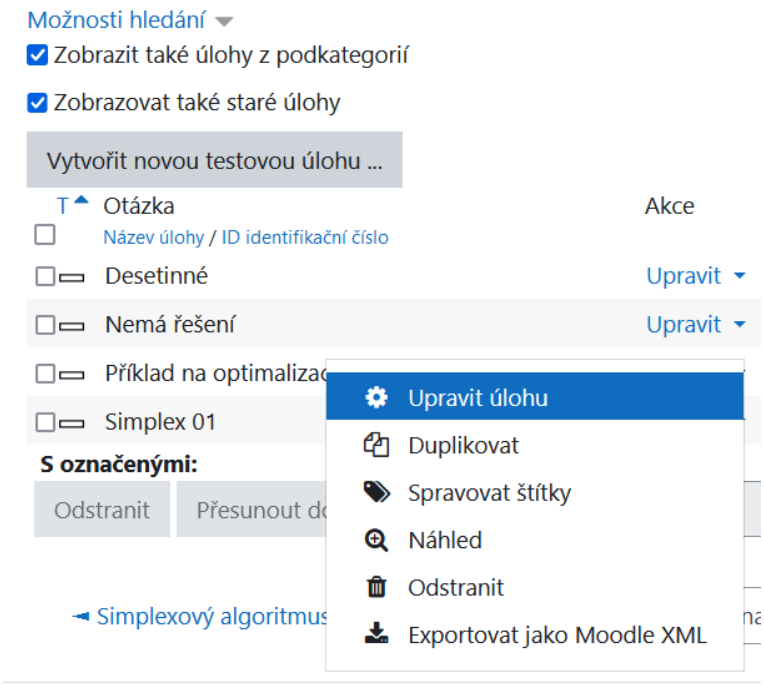
TeX Program pro počítačovou sazbu matematických výrazů
(anglicky)

API Rozhraní pro programování aplikací
(anglicky Application Programming Interface)

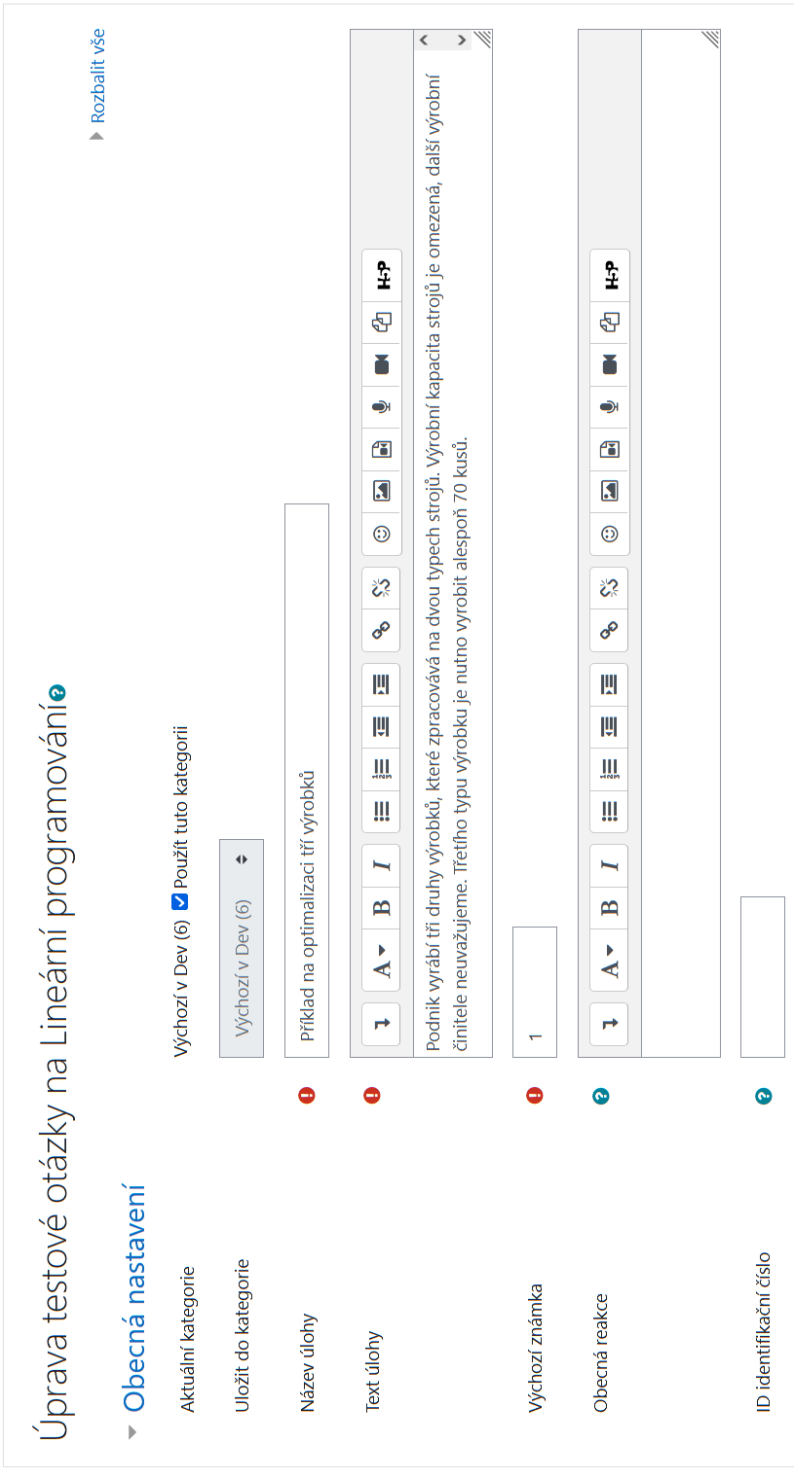
PHP Hypertextový preprocesor
(anglicky Hypertext Preprocessor)

Příloha **B**

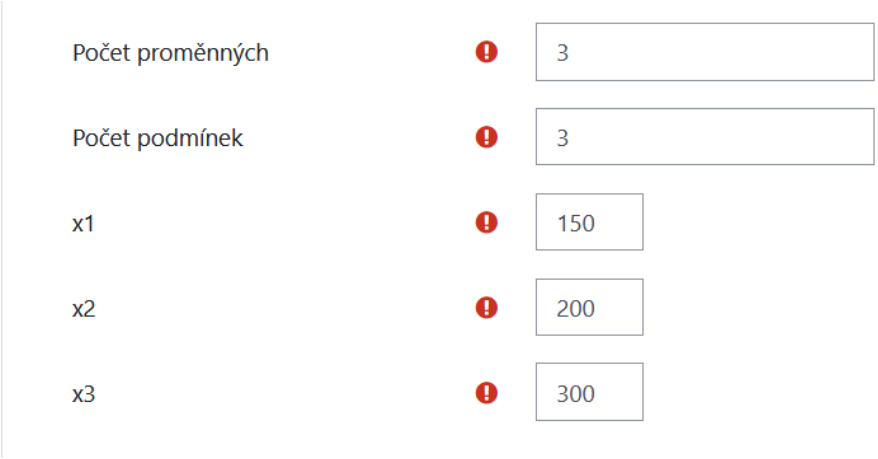
Přílohy snímků obrazovky



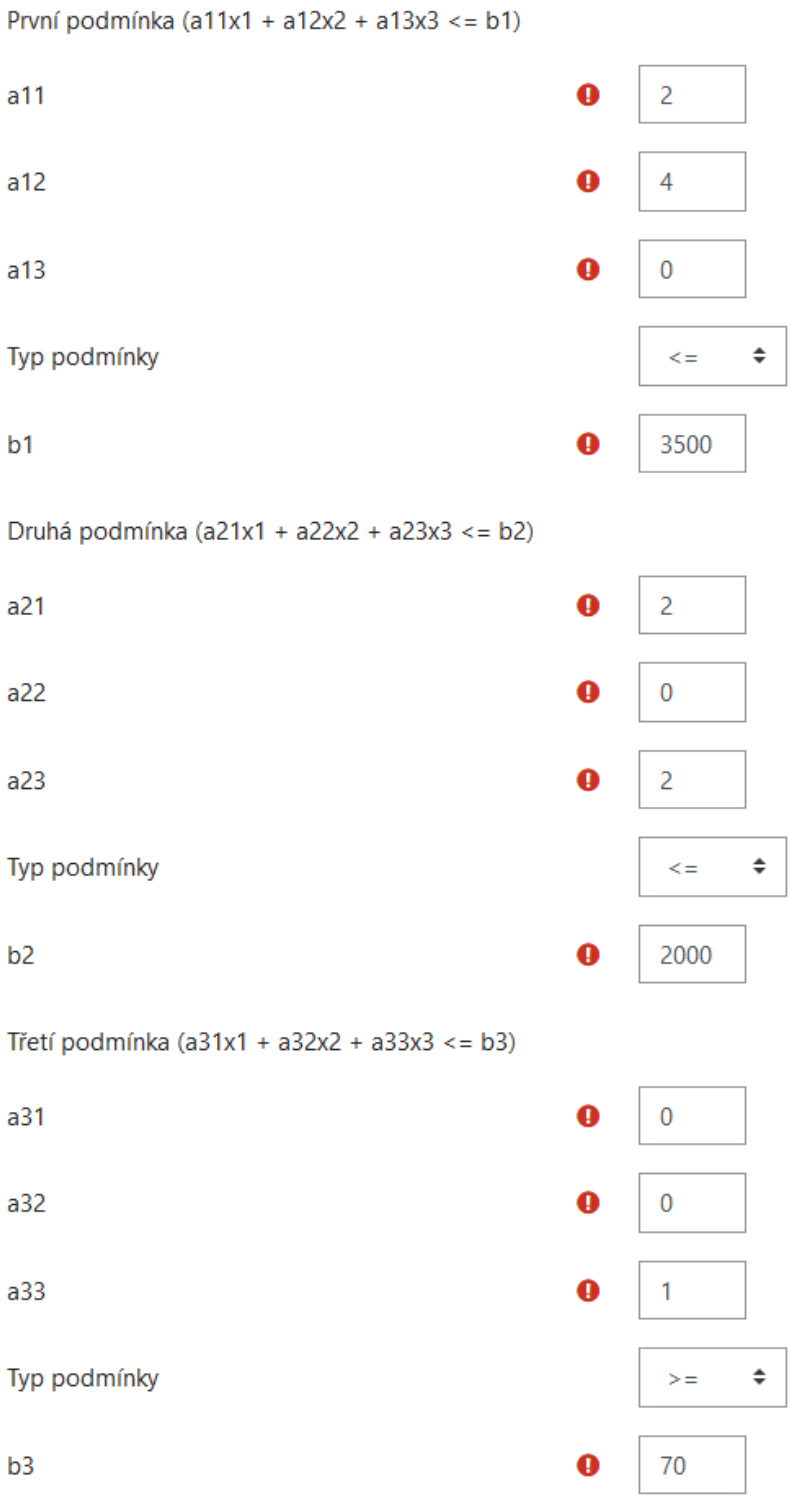
Obrázek B.1: Úprava testové otázky
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí



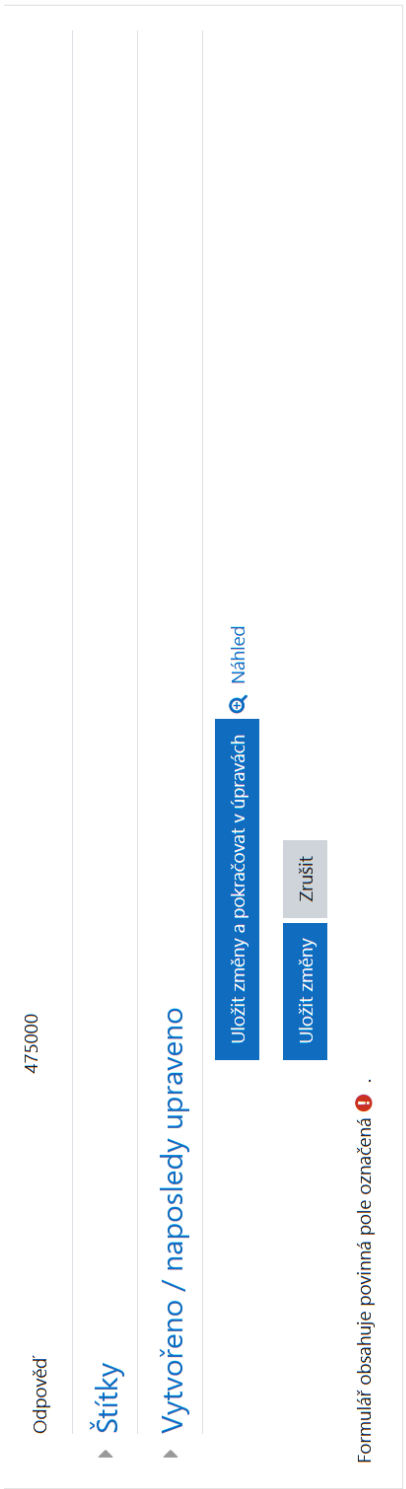
Obrázek B.2: Obecné nastavení při úpravě testové otázky
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí



Obrázek B.3: Nastavení proměnných a počtu podmínek při úpravě testové otázky
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí



Obrázek B.4: Nastavení omezujících podmínek při úpravě testové otázky
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí



Obrázek B.5: Vypočítaný výsledek při úpravě testové otázky
Zdroj: snímek obrazovky Bc. Denise Drdy z vývojového prostředí

Příloha **C**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `qtype_linprog_moodle.zip` .. zdrojové kódy implementace ve formátu ZIP a zároveň instalační soubor pro Moodle
    - `qtype_linprog_dokumentace.zip` ..... vygenerovaná dokumentace ve formátu ZIP
  - `thesis` ..... zdrojová forma práce ve formátu ZIP
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF