TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta přírodovědně-humanitní
a pedagogická

Programování v pregraduální přípravě učitelů informatiky

Diplomová práce

Studijní program: N1101 – Matematika
Studijní obory: 7504T077 – Učitelství informatiky pro střední školy2
7504T089 – Učitelství matematiky pro střední školy

Autor práce: **Bc. Ondřej Vraštil**
Vedoucí práce: Mgr. Jan Berki, Ph.D.

Liberec 2017

Technická univerzita v Liberci
Fakulta přírodovědně-humanitní a pedagogická
Akademický rok: **2015/2016**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Ondřej Vraštil**
Osobní číslo:	**P14000602**
Studijní program:	**N1101 Matematika**
Studijní obory:	**Učitelství informatiky pro střední školy**
	Učitelství matematiky pro střední školy
Název tématu:	**Programování v pregraduální přípravě učitelů informatiky**
Zadávající katedra:	**Katedra aplikované matematiky**

Zásady pro vypracování:

V souvislosti s probíhající diskusí kolem revize informatického kurikula na základních a středních školách je důležité vědět, jak jsou v současné době budoucí učitelé informatiky připravováni pro jedno ze ztěžejních témat – programování, resp. algorotmizaci. Cílem diplomové práce je na základě analýzy deklarovaného kurikula popsat obvyklý a doporučený koncept výuky programování v rámci pregraduální přípravy učitelů informatiky v České republice.

1. Student provede rešerši českých i zahraničních zdrojů týkajících se deklarovaného kurikula, pregraduální přípravy učitelů informatiky, kurikulárního výzkumu a konceptů výuky programování. Na základě rešerše popíše dosavadní stav zkoumané problematiky a vymezí hlavní pojmy.

2. Na základě výsledků předchozích výzkumů a vzhledem k cílům práce stanoví otázky, na základě nichž by bylo možné porovnat strukturu a obsah výuky programování pro danou cílovou skupinu.

3. Získá studijní plány v Česku akreditovaných studijních programů zaměřených na přípravu budoucích učitelů informatiky v rámci bakalářského a magisterského stupně. Dále pak sylaby jednotlivých předmětů především zaměřených na algoritmizaci a programování. Získané materiály analyzuje především pomocí textové analýzy.

4. Na základě výsledků popíše koncepty výuky programování a vymezí jejich zásadní rozdíly. Dále navrhne a zdůvodní vhodný koncept pro oba stupně pregraduální přípravy učitelů informatiky v dané oblasti.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **cca 70 stran**

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

Seznam odborné literatury:

- **BERKI, Jan.** *Study programs of training teachers of informatics in the Czech republic*. **In: Journal of Technology and Information Education [online]. 1/2013, Volume 5, Issue 1. s. 7580. ISSN 1803-6805. Dostupné z http://www.jtie.upol.cz/13_1.htm.**
- **SCHUBERT, Sigrid a Andreas SCHWILL.** *Didaktik der Informatik*. **2. vydání. Heidelberg: Springer Spektrum, 2011. ISBN 978-3-8274-2652-9.**
- **SATRAPA, Pavel.** *Pascal pro zelenáče*. **Praha: Neokortex, 2000. ISBN 80-86330-03-6.**
- **PECINOVSKÝ, Rudolf.** *OOP: naučte se myslet a programovat objektově*. **Brno: Computer Press, 2010. ISBN 978-80-251-2126-9.**

Vedoucí diplomové práce: **Mgr. Jan Berki**

Katedra aplikované matematiky

Datum zadání diplomové práce: **17. května 2016**

Termín odevzdání diplomové práce: **28. dubna 2017**

prof. RNDr. Jan Picek, CSc.
děkan

Technická univerzita v Liberci L.S. - 10 -

doc. RNDr. Miroslav Kouckỳ, CSc.
vedoucí katedry

V Liberci dne 17. května 2016

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Poděkování

Rád bych poděkoval dr. Janu Berkimu za vzor, kterým mi byl po celou dobu mého studia.

Anotace

Cílem diplomové práce je na základě analýzy deklarovaného kurikula popsat obvyklý a doporučený koncept výuky programování v rámci pregraduální přípravy učitelů informatiky v České republice. V první fázi výzkumu jsou v příslušných vzdělávacích programech vyhledány a kategorizovány všechny předměty související s tématem algoritmizace, programování a didaktiky programování. V další fázi jsou metodou konceptuální analýzy zkoumány vybrané skupiny předmětů, načež jsou jednotlivé studijní programy na základě této analýzy porovnávány. Poslední fází výzkumu je návrh ideálního konceptu výuky programování a didaktiky programování sestavený na základě posledních výzkumů v této oblasti.

Klíčová slova: programování, algoritmizace, koceptualní analýza, pregraduální příprava.

Annotation

The aim of the master´s thesis is to describe the usual and recommended concept of teaching of programming in undergraduate degree education of CS teachers in the Czech Republic, based on the analysis of the declared curriculum. In the first phase of the research, all subjects related to the topic of algorithmization, programming and didactics of programming are categorized. In the next phase, selected groups of subjects are examined by conceptual analysis, and the individual study programs are compared on the basis of this analysis. The last phase of the research is the design of an ideal concept of programming education and didactics of programming, based on the latest research in this field.

Keywords: Programming, Algorithmization, conceptual analysis, undergraduate education.

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

ISCED	International Standard Classification of Education
med.	medián
OOP	objektově orientované programování
prům.	průměr
RVP	rámcový vzdělávací program
SŠ	střední škola
ŠVP	štátný vzdělávací program
VŠ	vysoká škola
ZŠ	základní škola

1 Úvod

Nedílnou součástí pregraduální přípravy budoucích učitelů informatiky základních i středních škol je výuka programování a teorie algoritmů. Toto téma v současnosti získává na popularitě i na středních a základních školách, kde je mu věnováno stále více pozornosti. Výuka úvodu do programování a algoritmického myšlení je nejen obsažena v rámcovém vzdělávacím programu pro gymnázia a některé odborné střední školy, ale v uzpůsobené podobě se objevuje i ve výuce na školách základních, kde se mohou žáci setkat se speciálními programovacími jazyky pro děti.

Aby mohla být správně provedena didaktická transformace učiva směrem k žákovi, musí její původce-učitel mít dostatečný vhled do problematiky a patřičné porozumění tématu. Jednou z hlavních premis je kvalitní vzdělání v rámci pregraduální přípravy, ve které by měl student-budoucí učitel získat takové vzdělání, aby dokázal obstojně předat znalosti svým žákům. Protože je informatika mladý a dynamicky se rozvíjející obor, na kvalitu pregraduální přípravy má vliv i relevance a aktualita vyučovaných témat. V dnešní době se na našich základních a středních školách začínají využívat pro školní prostředí nová témata jako je robotika nebo unplugged teaching, pro která je znalost algoritmizace nutná. Fakulty připravující učitele informatiky by měly pružně reagovat na moderní trendy ve výuce a uspokojit poptávku studentů po kvalitním a hlavně využitelném vzdělání, které budou moci využít ve své budoucí praxi, nehledě na to, že z některých studentů se mohou stát i vysokoškolští pedagogové.

Jak si naše vysoké školy (dále jen VŠ) vedou ve výuce programování a algoritmizace? Následují moderní trendy a požadavky zaměstnavatelů budoucích absolventů? Je pořadí předmětů během studia smysluplné? Dá se vysledovat podobnost mezi programy napříč republikou? Existuje jeden nejvhodnější způsob, jak učit programovaní budoucí učitele, nebo je možné volit z více cest?

Abychom tyto otázky mohli zodpovědět, je v první řadě nutné pokusit se analyzovat zdroje týkající se programování a pregraduální přípravy učitelů. Dále je potřeba získat data o obsahu jednotlivých předmětů v rámci pregraduální přípravy napříč fakultami v ČR. Fakulty tato data zveřejňují v sylabech, které jsou volně k dispozici na stránkách jednotlivých fakult. Získaná data budou zkoumána formou textové analýzy, jež by mohla na výše zmíněné otázky odpovědět. Na výsledcích teoreticko-rešeršní i praktické části bude postavena závěrečná kapitola, návrh vhodného konceptu výuky programování na VŠ pro budoucí učitele středních i základních škol.

2 Programování

Pokud máme zkoumat pregraduální přípravu učitelů informatiky v oblasti programování, měli bychom nejdříve odpovědět na důležitou otázku – Jak definuje odborná literatura programování a algoritmizaci? Dále je třeba zkoumat, zda jsou tato témata zakotvena v rámcových vzdělávacích programech základních a středních škol v ČR a zahraničí, abychom zjistili, zda se budoucí učitel s těmito tématy může setkat. Nakonec na základě zkoumání odborných zdrojů odpovíme na dvě důležité otázky: Proč by vlastně mělo být programování a algoritmizace součástí pregraduální přípravy budoucích učitelů informatiky na VŠ? Jakým způsobem mohou být tato témata vyučována?

2.1 Algoritmus a algoritmizace – definice pojmů

> *„Before there were computers, there were algorithms. But now that there are computers, there are even more algorithms, and algorithms lie at the heart of computing.“*
>
> *Introduction to Algorithms, Thomas H. Cormen et al.*

„Algoritmus je jeden z ústředních (ne-li ten vůbec nejústřednější) pojem z oblasti informatiky. (Schubert – Schwill 2011)“ V literatuře najdeme několik jeho definic. Schubert a Schwill (2011, s. 4) nahlíží na algoritmus jako na „formálními prostředky popsatelný, mechanicky proveditelný postup k řešení třídy problému“. Cormen (2013, s. 1) popisuje algoritmus jako „sadu kroků ke splnění úlohy“, počítačový algoritmus jako „sadu kroků ke splnění úlohy tak přesně popsaných, aby je dokázal vykonat počítač“. Pro Skienu (2008, s. 3) je algoritmus „procedura ke splnění konkrétní úlohy a myšlenka, která stojí za počítačovým programem“. Cormen, Leiserson, Rivest a Stein (2009, s. 5) ve své obsáhlé publikaci pojímají algoritmus jako „jasně definovanou výpočetní proceduru, která přijímá hodnotu nebo soubor hodnot jako vstup a produkuje hodnotu nebo soubor hodnot jako výstup“. Známý informatik Donald Knuth (2008, s. 5), tvůrce typografického systému Tex, definuje algoritmus jako „konečnou množinou pravidel, která popisují posloupnost operací pro řešení jistého typu problémů“. Další definici najdeme u Harela a Feldmana (2004, s. XII): „Algoritmus je abstraktní návod předepisující proces, který by mohl být proveden člověkem, počítačem nebo jinými prostředky.“ Podle Knutha (2008, s. 4–6) musí algoritmus splňovat několik základních vlastností:

- *konečnost* – algoritmus musí vždy po určitém počtu kroků skončit;
- *určitost* – každý krok algoritmu musí být přesně definován a pro každý případ v něm musí být s určitostí a jednoznačností popsány prováděné operace;
- *vstup* – každý algoritmus má nula nebo více vstupů: to jsou veličiny, které do algoritmu zadáme před jeho zahájením, nebo které načteme dynamicky za běhu;
- *výstup* – algoritmus má také jeden nebo více výstupů: to jsou veličiny, které mají zadaný vztah ke vstupům;
- *efektivita* – algoritmus by měl být zároveň efektivní, což znamená, že všechny jeho operace musí být v rozumné míře jednoduché, takže by je v principu měl být schopen přesně a za konečnou dobu provést kdokoli s tužkou a papírem.

Často najdeme přirovnání algoritmu ke kuchařskému receptu jako například u Knutha (2008, s. 6) nebo u Harela a Feldmana (2004, s. 4). Vstupem jsou pak v tomto případě suroviny, výstupem je hotové jídlo. Recept je konečný, naše vaření nebude probíhat nekonečně dlouho a efektivní – můžeme je provést v relativně krátkém čase.

Proces převodu problému na jednotlivé kroky nazýváme **algoritmizace** (Schubert – Schwill 2011, s. 67). Motyčka (1999, s. 5) chápe algoritmizaci problému při tvorbě programu jako vytváření postupu řešení daného problému na počítači, dále jako

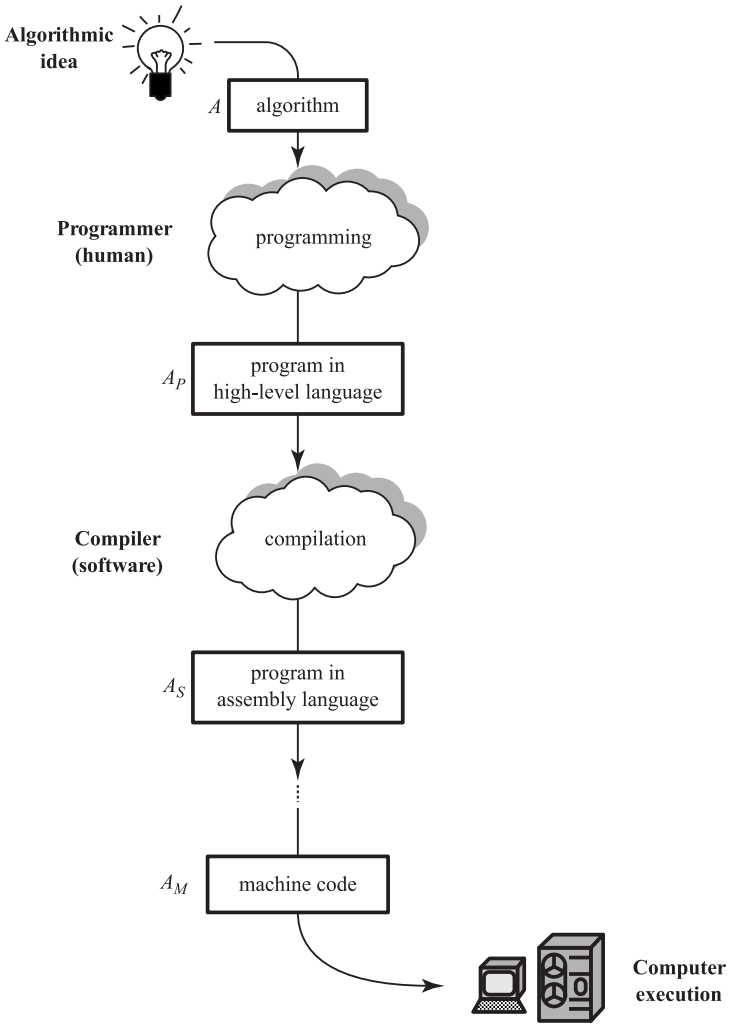
> „krásnou tvůrčí činnost, při které využíváme intelekt, zkušenosti, intuici a postupně vytváříme spoustu postupů řešení, z nichž ty počáteční zpravidla k cíli nevedou vůbec, další vedou k cíli pouze občas a havárie nastává již jen v určitých zvláštních (mezních) situacích, o kterých jsme na počátku našeho snažení vůbec neuvažovali."

Jak je z výše uvedených definic patrné, pojmy algoritmus a algoritmizace jsou důležitou součástí informatiky, ale můžeme říci, že ji i přesahují, když algoritmem můžeme popsat postup počítačům vzdálený – například proces vaření. Pro potřeby této práce je potřeba správně rozhodnout, které předměty v průběhu pregraduální přípravy pomáhají studentovi v rozvíjení jeho schopnosti algoritmizace problémů, protože to v sylabech těchto předmětů nemusí být explicitně zmíněno – stejně tak, jako když se při výuce vaření na základní škole (dále jen ZŠ) o algoritmizaci nezmiňujeme, ačkoliv se o algoritmizaci jedná. Na základě výše uvedených definic lze konstatovat, že na algoritmus se dá nahlížet dvěma způsoby – první uvádí algoritmus v úzkém vztahu s informatikou a spojuje ho s pojmy jako „počítačový program" nebo „výpočetní proceruda", druhý ho popisuje spíše jako sadu kroků k řešení blíže nespecifikovaného problému. První zde popsaný pohled je tak konkrétnější variantou toho druhého, jímž bude nahlíženo na algoritmus v této práci.

2.2 Programování a jeho vztah k algoritmizaci

Poté, co byl definován pojem algoritmus, je nutné definovat i jeho propojení s teorií počítačů – strojů. Počítače v současné době slouží k řešení mnoha problémů okolo

nás, např. automatickému pilotování letadla, nebo složitým simulacím chemických procesů (Harel – Feldman 2004, s. 49). Vše jsou složité algoritmy vykonávané počítačem. Aby počítač mohl provést příkaz, je algoritmus zapsán v **programovacím jazyku**, umělém jazyku, který je zpravidla zredukovanou podmnožinou anglického jazyka. Programem pak rozumíme zápis algoritmu v programovacím jazyku (Motyčka 1999, s. 6). Jazyky se liší mírou abstrakce – vyšší jazyky se podobají lidské řečí, jsou více abstraktní, naopak pomocí nižších jazyků je možné lépe ovládat strojové operace, mají nižší abstrakci. Schéma na obrázku 2.1 popisuje celý princip od myšlenky algoritmu až po jeho provedení počítačem (Harel – Feldman 2004, s. 56).



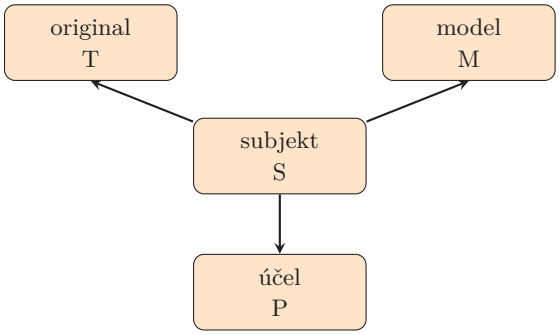
Obrázek 2.1: Proces provedení algoritmu počítačem

Programátor-člověk naprogramuje algoritmus do programu v některém z vyšších jazyků. Poté si počítač převádí, tzv. kompiluje tento program do nižšího jazyku, jazyku symbolických adres (anglicky *assembly language*), který už přímo popisuje místa v počítačové paměti. Následně je program převeden do strojového kódu, jehož instrukcím rozumí procesor a je schopen je provést. Alternativa ke kompilaci je převod pomocí interpretru. Interpretr využívá výpočetní zdroje za běhu programu,

aniž by by bylo nutné program kompilovat (Harel – Feldman 2004, s. 54–57).

2.3 Programovací paradigmata

V jistém smyslu můžeme na informatiku nahlížet jako na vědu, která se zabývá vytvářením realizovatelných modelů. Proces vytváření modelů se dá popsat jako relace *R(S,P,T,M)*, kde subjekt S vytváří model M podle originálu T za účelem P (Schubert – Schwill 2011, s. 135). Jelikož pomocí počítače modelujeme mnoho různých situací, bylo vytvořeno mnoho programovacích jazyků s různými specifiky. Jazyky můžeme kategorizovat do několika hlavních stylů, které se liší přístupem k abstrakci dat a operacím s nimi, a představuji specifický přístup k programu a jeho provedení. Takovéto styly nazýváme programovací paradigmata (Bolshakova 2005, s. 286).



Obrázek 2.2: Schéma vytváření modelů

Imperativní paradigma se vyvinulo z nižších jazyků – strojového a jazyku symbolických adres (Bolshakova 2005, s. 286). Z pohledu tohoto paradigmatu je počítač soubor paměťových buněk organizovaných do různých typů datových struktur, jako pole nebo seznam. Program v tomto paradigmatu je sekvence přesných příkazů, která mění data v těchto strukturách a které jsou spuštěny v přesném sledu. Jelikož program pracuje se stále stejným paměťovým místem, ke kterému uživatel přistupuje pomocí tzv. proměnných, měl by mít programátor v každém kroku programu přehled, jak se tyto hodnoty mění.

Ve **funkcionálním paradigmatu** je programem výraz, který se skládá z funkcí s podobnou strukturou, jakou mají funkce matematické. Jedna funkce může být argumentem jiné funkce. Po spuštění proběhne zjednodušení tohoto výrazu až do podoby, kdy zjednodušit dále nelze. Neexistuje zde princip modifikovatelné paměti jako v imperativním paradigmatu (Škavrda 2006).

V **logickém paradigmatu** je program brán jako soubor logických formulí – axiomů, které popisují vztahy a vlastnosti nějakých objektů, a dotaz, který má být na základě těchto pravidel zodpovězen – dokázán (Skoupil 2007b).

Základní myšlenka **objektově orientovaného paradigmatu** se zakládá na představě, že reálný svět je množství objektů, které spolu navzájem komunikují. Definujeme zde skupiny prvků s podobnými vlastnostmi, jež nazýváme třídy. Na

rozdíl od imperativního paradigmatu, ve kterém jsou funkce považovány za aktivního činitele modifikujícího pasivně uložená data, objektově orientované paradigma považuje za aktivní v paměti uložené objekty a za pasivní zprávy, které si objekty mezi sebou posílají.

Přehlednou tabulku paradigmat uvádí Bolshakova (2005, s. 287):

Tabulka 2.1: Přehled programovacích paradigmat

Paradigma	**Klíčový koncept**	**Program**	**Provedení programu**	**Výsledek**
Imperativní	Příkaz (instrukce)	Sekvence příkazů	Provedení příkazů	Zavěrečný stav počítačové paměti
Funkcionální	Funkce	Soubor funkcí	Vyhodnocení funkcí	Hodnota hlavní funkce
Logické	Predikát	Logické formule: axiomy a theorémy	Logické dokazování theorému	Úspěch nebo neúspěch dokazování
Objektově orientované	Objekt	Soubor tříd a objektů	Výměna zpráv mezi objekty	Finální stav objektových stavů

Pro úplnost uveďme ještě tabulku, s příklady programovacích jazyků pro příslušná paradigmata. Tato tabulka samozřejmě není úplná, pro každé paradigma najdeme nespočet jazyků, uvádí pouze základní přehled. V tabulce jsou uvedeny i jazyky, ve kterých můžeme programovat v několika různých paradigmatech, takové jazyky nazýváme **multiparadigmatické**.

Tabulka 2.2: Příklady programovacích jazyků v jednotlivých paradigmatech

Paradigma	**Programovací jazyk**
Imperativní	C, C++, Java, PHP, Python, Ruby
Funkcionální	Python, Lisp, Scheme
Logické	Prolog
Objektově orientované	C++, Java, Python

Je součástí pregraduální přípravy učitelů v ČR výuka všech paradigmat, nebo se studium zaměřuje pouze na některá? S kterým paradigmatem se seznámí studenti nejdříve? Jsou pro jednotlivá paradigmata vyhrazeny samostatné předměty, nebo je koncentrovaná výuka více paradigmat do jednoho předmětu? To jsou otázky, které zodpovíme ve výzkumné části.

2.4 Programování z pohledu deklarovaného kurikula ZŠ a SŠ

Jedním z určujících dokumentů pro vzdělávání na základních a středních školách je rámcový vzdělávací program. Tento kurikulární dokument určuje a specifikuje obsah výuky na republikové úrovni a odvíjí se od něho dokumenty na úrovni nižší – školní vzdělávací programy, má tedy zásadní vliv na podobu výuky informatiky v republice. Reflektuje pregraduální příprava obsah těchto klíčových dokumentů? Analyzujme nyní tento dokument, abychom zjistili, zda a jak jsou v něm témata algoritmizace a programování obsažena.

V RVP pro základní vzdělávání (MŠMT 2016) spadá výuka programování a algoritmizace pod tzv. vzdělávací oblast *Informační a komunikační technologie*. Charakteristika oblasti pojem algoritmus, programování ani jím příbuzná nebo odvozená slova neuvádí. Zaměřuje se spíše na výuku práce s výpočetní technikou a práci s informací. Zmínku najdeme v takzvaných cílových zaměřeních, kdy je uvedeno, že: „vzdělání v oblasti vede žáka k schopnosti formulovat svůj požadavek a využívat při interakci s počítačem algoritmické myšlení“. RVP dále kategorizuje vzdělávací obsah a rozděluje ho do učiva prvního a druhého stupně, pojem algoritmizace ani programování zmiňován není.

V RVP pro gymnázia se mění (rozšiřuje) název vzdělávací oblasti na *Informatika a informační a komunikační technologie*. Jak název napovídá, součástí výuky na gymnáziích by měly být základy informatiky jako vědy. Jeho charakteristika se zaměřuje se hlavně na způsob myšlení, kdy „cílem je zpřístupnit žákům základní pojmy a metody informatiky, napomáhat rozvoji abstraktního, systémového myšlení, podporovat schopnost vhodně vyjadřovat své myšlenky, smysluplnou argumentací je obhajovat a tvůrčím způsobem přistupovat k řešení problémů.“ Pojem algoritmus se tu objevuje už explicitně také: „Žák se seznámí se základními principy fungování prostředků ICT a soustředí se na pochopení podstaty a průběhu informačních procesů, algoritmického přístupu k řešení úloh a významu informačních systémů ve společnosti.“ I mezi cíli je algoritmizace zastoupena, a to přímo jako bod *Uplatňování algoritmického způsobu myšlení při řešení problémových úloh*. Ve vzdělávacím obsahu se pak objevuje tematický okruh *Zpracování a prezentace informací*, kdy jeden z očekávaných výstupu uvádí, že žák „aplikuje algoritmický přístup k řešení problémů.“ To reflektuje i popis očekávaného učiva a jeho bod *algoritmizace úloh – algoritmus, zápis algoritmu, úvod do programování*, na gymnáziu je tedy výuka programování a algoritmizace **povinnou součástí**, ale jak vzdělávací cíle, tak očekávané učivo jsou popsány velmi obecně.

Vzdělávání na odborných středních školách probíhá také podle RVP, každý obor má svůj vlastní dokument. Analyzujme nyní výskyt algoritmizace a programování na RVP oboru Informační technologie (MŠMT 2008), kde by úroveň informatického vzdělávání měla být nejrozsáhlejší z celého středoškolského systému. Programování najdeme v tzv. klíčových odborných kompetencích, konkrétně je zde uvedenáí konkrétní kompetence *Programovat a vyvíjet uživatelská, databázová a webová řešení*, podle níž je cílem, aby absloventi:

- algoritmizovali úlohy a tvořili aplikace v některém vývojovém prostředí;
- realizovali databázová řešení;
- tvořili webové stránky.

V RVP pro odborné vzdělávání existují vzdělávací oblasti tak jako v RVP pro gymnázia, dělí se ještě dále na tzv. vzdělávací okruhy, podle kterých se na školní úrovni definuje obsah jednotlivých předmětů. Pro programování má odborné vzdělávání samostatný okruh nazvaný *Programování a vývoj aplikací* jehož cílem je „naučit žáka vytvářet algoritmy a pomocí programovacího jazyka zapsat zdrojový kód programu“. V tabulce (2.3) obsažené v RVP najdeme definované výsledky vzdělávání a učivo. Díky analýze jsme tak zjistili, že pro odborné školy je vzdělávací obsah blíže specifikován než v RVP pro gymnázia, například zde najdeme i zmínky o jednotlivých programovacích paradigmatech.

Tabulka 2.3: Obsahový okruh *programování a vývoj aplikací*

Výsledky vzdělávání	Učivo
Žák: • zná vlastnosti algoritmu; • zanalyzuje úlohu a algoritmizuje ji; • zapíše algoritmus vhodným způsobem;	**1 Algoritmizace** • význam, prvky algoritmu
• použije základní datové typy; • použije řídící struktury programu; • vytvoří jednoduché strukturované programy;	**2 Strukturované programování** • datové typy • řídicí struktury
• rozumí pojmům třída, objekt a zná jejich základní vlastnosti; • použije jednoduché objekty;	**3 Úvod do objektového programování** • třída, objekt, vlastnosti tříd
• zná výhody použití jazyka SQL; • použije základní příkazy jazyka SQL;	**4 Základy jazyka SQL** • základní příkazy (SELECT, UPDATE, INSERT, DELETE)
• aplikuje zásady tvorby WWW stránek; • orientuje se ve struktuře HTML stránky; • vytvoří webové stránky včetně optimalizace a validace; • použije formuláře a skriptovací jazyk.	**5 Tvorba statických a dynamických webových stránek**

Shrňme si, jak je programování a algoritmizace obsažena v RVP všech stupňů vzdělávání a jaký to má důsledek na výuku. Celý koncept RVP dává škole pouze obecný rámec, který by měla dodržovat. Pro jednotlivá témata zde není uvedena časová dotace, což dává možnost upřednostnit některá témata před ostatními. Dále zde nejsou uvedeny metody, které mají být při výuce použity, takže např. algoritmizace může být procvičována mnoha různými způsoby. Pro programování nejsou na národní úrovni určeny ani doporučeny konkrétní programovací jazyky, kromě odborného vzdělávání není ani určeno, jaké paradigma by mělo být při výuce použito. Mezi základními školam mohou vznikat značné rozdíly, kdy některé ŠVP mohou zahrnovat programování (skrze dětské programovací jazyky) na prvním stupni, ně-

které vůbec programování zařazovat do výuky nemusí.[1] Obecně lze říci, že výuka ICT je směřována spíše k ovládání informačních a komunikačních technologií, než k práci s algoritmizací.

Pokud by měla být pregraduální příprava tvořena jen a pouze podle očekávaných výstupů v RVP, v programu pro ZŠ by se výuka programování nemusela objevit vůbec, stačila by výuka algoritmizace. V programu pro SŠ už se programování objevit musí, nejsme ale omezeni konkrétním programovacím jazykem, pouze by měla být vyučována průprava jak do strukturovaného, tak i objektově orientovaného paradigmatu. Takováto pregraduální příprava by ale samozřejmě nedávala studentům VŠ dostatečný vhled do problematiky a nepřipravila by je na možné budoucí změny v RVP. Můžeme ale konstatovat, že podoba RVP dává fakultám velkou volnost při přípravě programů pro budoucí učitele informatiky. Fakulty tak mohou do svých programů reflektovat, jak podle nich vypadá kvalifikovaný učitel informatiky.

Deklarované kurikulum výuky informatiky na Slovensku

Jelikož je informatika mladý obor a změny v něm rychlé, kurikulární dokumenty nemusí obsahovat nejaktuálnější trendy v oboru. Je proto vhodné sledovat i vývoj zahraničního přístupu k výuce informatiky a porovnávat s tím tuzemským. Podobnou struktru jako české RVP mají i kurikulární dokumenty Slovenské republiky – štátné vzdělávácí programy (dále jen ŠVP) z roku 2008 (ŠPÚ 2008), svým obsahem se ale díky posledním reformám liší. Porovnání těchto kurikulárních dokumentů pro úroveň ISCED 1 a 2 provedl Berki (2011), nedostatečná specifikace vzdělávacího obsahu a absence práce s informatikou jako vědou a algoritmickým myšlením v českém RVP byly jeho hlavní zjištění (Berki 2011, s. 36). Obsah učiva informatiky byl v slovenském ŠVP pro školy úrovně ISCED 1–3 rozdělen do pěti okruhů:

- Informácie okolo nás;
- Komunikácia prostredníctvom IKT;
- Postupy, riešenie problémov, algoritmické myslenie;
- Princípy fungovania IKT;
- Informačná spoločnosť;

Algoritmizaci se věnuje okruh *Postupy, riešenie problémov, algoritmické myslenie*, ve kterém jsou dále definovány obsahové a výkonové standardy[2]. Jako příklad uveďme tabulku 2.4, která popisuje tento okruh pro úroveň ISCED3.

[1] Měl jsem osobní zkušenost na dvou základních školách, jedna s využitím disponibilních hodin vyučovala informatiku během 3.–7. ročníku, kdy už ve třetím ročníku výuka obsahovala dětský programovací jazyk Baltík. Druhá škola měla nejnižší možnou dotaci jednu hodinu pro každý stupeň a o výuce programování se zde vůbec neuvažovalo, jednalo spíše o výuku informačních technologií.

[2] Obsahovým standardem rozumíme obsah probíraného učiva, výkonovým standardem jsou výstupní kompetence absolventa.

Tabulka 2.4: Obsahový okruh *Postupy, riešenie problémov, algoritmické myslenie*

Výkonový štandard	Obsahový štandard
• Analyzovať problém, navrhnúť algoritmus riešenia problému, zapísať algoritmus v zrozumiteľnej formálnej podobe, overiť správnosť algoritmu. • Riešiť problémy pomocou algoritmov, vedieť ich zapísať do programovacieho jazyka, hľadať a opravovať chyby. • Rozumieť hotovým programom, určiť vlastnosti vstupov, výstupov a vzťahy medzi nimi, vedieť ich testovať a modifikovať. • Riešiť úlohy pomocou príkazov s rôznymi obmedzeniami použitia príkazov, premenných, typov a operácií. • Používať základné typy používaného programovacieho jazyka • Rozpoznať a odstrániť syntaktické chyby, opraviť chyby vzniknuté počas behu programu, identifikovať miesta programu, na ktorých môže dôjsť k chybám počas behu programu.	• Problém. Algoritmus. Algoritmy z bežného života. Spôsoby zápisu algoritmov. • Etapy riešenia problému – rozbor problému, algoritmus, program, ladenie. • Programovací jazyk – syntax, spustenie programu, logické chyby, chyby počas behu programu. Pojmy – príkazy (priradenie, vstup, výstup), riadiace štruktúry (podmienené príkazy, cykly), premenné, typy , množina operácií.

Slovenské deklarované kurikulum má tedy pro algoritmizaci vlastní okruh, ve kterém jsou učivo i výstupní standardy rozepsány mnohem detailněji než v kurikulu českém. U programovacího jazyka jasně uvádí, které pojmy by měl žák znát. Stejně ale jako české RVP dává prostor k případné variaci, nedefinuje paradigma programovacího jazyka, ani blíže nespecifikuje programovací jazyky. Jelikož je dodržena jednotná struktura napříč všemi stupni vzdělávání, můžeme snadno identifikovat posuny v úrovni vzdělávání pro jednotlivá témata (Berki 2016, s. 85).

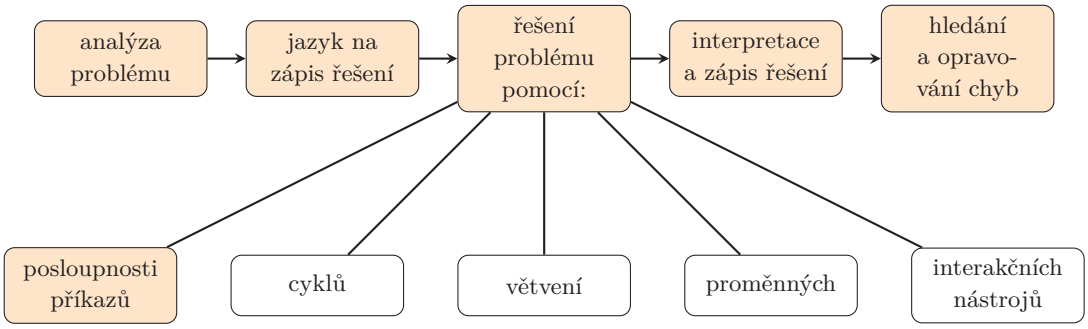
Slovensko ale v inovaci pokračovalo a v roce 2015 nasadilo inovovaný štatný vzdělávací program (ŠPÚ 2015), který všechny oblasti ještě dále specifikuje a rozděluje na tématické celky (ŠPÚ 2015). Uveďme nyní jako příklad několik těchto tématických celků s obsahovými a výkonovými standardy předmětu informatika pro úroveň ISCED3 okruhu *Postupy, riešenie problémov, algoritmické myslenie* pro 5.–8. ročník víceletého gymnázia[3]:

[3] Na rozdíl od dokumentu z roku 2011 je obsah vzdělávání rozdělen na dva celky, 1.–4. a 5.–8. ročník

Tabulka 2.5: Obsahový okruh *Programování a vývoj aplikací*

Výkonový štandard	Obsahový štandard
Analýza problému	
Žiak vie/dokáže • identifikovať vstupné informácie zo zadania úlohy, • popísať očakávané výstupy, výsledky, akcie, • identifikovať problém, ktorý sa bude riešiť algoritmicky, • formulovať a neformálne (prirodzeným jazykom) vyjadriť ideu riešenia, • uvažovať o vlastnostiach vykonávateľa (napr. korytnačka, grafické pero, robot, a pod.), • naplánovať riešenie úlohy ako postupnosť príkazov vetvenia a opakovania	Vlastnosti a vzťahy: zadaný problém – vstup – výstup Procesy: rozdelenie problému na menšie časti, syntéza riešenia z riešení menších častí, identifikovanie opakujúcich sa vzorov, identifikovanie miest pre rozhodovanie sa (vetvenie a opakovanie), identifikovanie všeobecných vzťahov medzi informáciami • význam, prvky algoritmu
Jazyk na zápis riešenia	
• používať jazyk na zápis algoritmického riešenia problému (použiť konštrukcie jazyka, aplikovať pravidlá jazyka), • rozpoznať a odstrániť chyby v zápise, • vytvárať zápisy a interpretovať zápisy podľa nových stanovených pravidiel (syntaxe) pre zápis algoritmov.	Pojem: program, programovací jazyk Vlastnosti a vzťahy: zápis algoritmu a vykonanie programu, vstup – vykonanie programu – výstup/akcia Procesy: zostavenie programu, identifikovanie, hľadanie, opravovanie chýb
Pomocou postupnosti príkazov	
• riešiť problém skladaním príkazov do postupnosti, • aplikovať pravidlá, konštrukcie jazyka pre zostavenie postupnosti príkazov.	Pojmy: príkaz, parameter príkazu, postupnosť príkazov Vlastnosti a vzťahy: ako súvisia príkazy a výsledok realizácie programu Procesy: zostavenie a úprava príkazov, vyhodnotenie postupnosti prí- kazov, úprava sekvencie príkazov (pridanie, odstránenie príkazu, zmena poradia príkazov)
Pomocou nástrojov na interakciu	
• rozpoznávať situácie, kedy treba získať vstup, • identifikovať vlastnosti vstupnej informácie (obmedzenia, rozsah, formát), • rozpoznávať situácie, kedy treba zobraziť výstup, realizovať akciu, • zapisovať algoritmus, ktorý reaguje na vstup, • vytvárať hypotézu, ako neznámy algoritmus spracováva zadaný vstup, ak sú dané páry vstup–výstup/akcia.	Vlastnosti a vzťahy: prostriedky jazyka pre získanie vstupu, spracovanie vstupu a zobrazenie výstupu Procesy: čakanie na neznámy vstup – vykonanie akcie – výstup, ná- sledný efekt

Výhodou je, že toto dělení je zachováno pro všechny stupně vzdělávání, snadno se identifikuje posun žáků v jejich znalostech a dovednostech, jak píše Berki (2016, s. 85), který uvádí i tabulku posunu pro jednotlivé celky. Pro náš celek Algoritmické řešení problému je zajímavé také to, že tématické celky úzce kopíruje samotný algoritmus vývoje softwaru, můžeme tyto celky zobrazit v přehledném diagramu (obrázek 2.3), barevně jsou označené bloky společné pro všechny úrovně vzdělávání ISCED1–ISCED3:



Obrázek 2.3: Schéma tématických celků

V roce 2016 uvedl Štátny pedagogický ústav metodické usměrnění (ŠPÚ 2016), kterým jasně popisuje svůj postoj k výuce programování:

> „Štátny vzdelávací program neurčuje konkrétny softvér a programovací jazyk, ktorý sa má vo vyučovaní informatiky používať. Je v kompetencii školy vybrať si programovací jazyk na vyučovanie na všetkých troch stupňoch vzdelávania vrátane maturitnej skúšky. Pre vyššie sekundárne vzdelávanie (vrátane maturantov) odporúčame (nie je to však povinné) jazyk Pascal alebo Python. Jazyk Pascal môže byť vo vývojovom prostredí Lazarus alebo FreePascal.“

Slovenské deklarované kurikulum tak nespecifikuje konkrétní programovací jazyk (některé z nich ale doporučuje) nebo programovací paradigma. Na rozdíl od českého ale specifikuje a rozděluje obsah vzdělávání do přehledných kategorií, úroveň dosaženého vzdělání je tím lépe ověřitelná.

Pokud budeme předpokládat, že jedním z determinantů pregraduální přípravy je i deklarované kurikulum na národní úrovni, slovenská verze kurikula má se svojí specifikovanější podobou výhodu. Tím, že slovenské kurikulum vymezilo pro algoritmizaci a programování vlastní okruh, dává tomuto tématu vysokou důležitost a bere ho jako nedílnou součást výuky informatiky na základních a středních školách.

2.5 Proč učit programování?

Programování je běžně součástí pregraduální přípravy učitelů (Berki 2013), toto téma ale nemá u studentů velkou oblibu (Fojtík 2015). Studenti mají obtíže předměty zaměřené na programování absolvovat, a proto by se raději ve své budoucí pedagogické praxi výuce programování vyhnuli (Fojtík 2015, s. 54). Studenti se staví spíše proti výuce programování na základních školách, má být podle nich součásti jen výběrových předmětů na gymnáziích. Uvádí důvody jako:

- výuka programování je příliš složitá pro žáky základních i většiny středních škol,
- programovací jazyky jsou nesrozumitelné a žáci by je nezvládli,
- vývojové nástroje jsou komplikované a nepřehledné,
- výuka programování je časově náročná a nezůstal by čas na důležitější témata,
- studenti mají negativní zkušenost s výukou programování na střední nebo základní škole,
- nepotřebujeme tolik programátorů v praxi.

Tento názor je patrně důsledkem jejich vlastní negativní zkušenosti s programováním ze ZŠ nebo SŠ. Takovýmto studentům je potřeba objasnit několik důvodů, proč by mohlo a mělo být programování a algoritmizace součástí kurikula jak na VŠ, tak na nižších stupních (Fojtík 2015, s. 54). Výuka programování a algoritmizace není příprava na povolaní programátora, ale „cílem programování ve škole je rozvoj tvořivosti a myšlení. Samotné programování je (skvělým) nástrojem k dosahování těchto cílů“ (Lessner 2015). Koncepty, které se studenti naučí během odborného výkladu, mohou na základních a středních školách předávat způsobem, kterému rozumí děti, např. pomocí dětských programovacích jazyků jako Scratch nebo Logo, případně při výuce robotiky.

Schubert a Schwill (2011) označili jako základní myšlenky informatiky pojmy algoritmus, jazyk a strukturální rozklad. Programování s těmito pojmy úzce souvisí, dá se tak označit za jedno z důležitých témat informatiky a mělo by tak být součástí oboru, který připravuje budoucí učitele informatiky.

Ačkoliv v našem RVP má algoritmizace a programování oproti uživatelsky zaměřeným informačním technologiím jen malé zastoupení, v zahraničí se situace v poslední době mění. „V současné době probíhají v řadě zemí kurikulární reformy, v nichž se vymezuje a mění postavení informatiky. Kurikulární reforma, v níž má významné postavení informatická složka, probíhá například v Polsku, v Austrálii nebo v Rusku.“ (Černochová – Vaníček 2015). Na Slovensku se setkávají žáci s informatickými tématy jako algoritmické myšlení, procedury a princip fungování digitálních technologii už od 3. třídy. V Anglii výuka informatiky probíhá ve specializovaném předmět Computing, jehož cílem je rozumět a aplikovat základním principům informatiky a v jehož rámci tvoří žáci jednoduché programy už od 1. třídy (Department for Education 2013). Je možné, že se dočkáme v ČR změny kurikulárních dokumentů tímto směrem, studenti učitelství by na to měli být připravováni.

Velkým příznivcem programování ve vzdělávání byl Seymour Papert, tvůrce dětského programovacího jazyku LOGO. Ve své knize Mindstorms (Papert 1993) po-

pisuje mnoho pozitivních dopadů, které mají počítače a programování na vývoj a vzdělávání dětí. Ačkoliv tato kniha vyšla už v roce 1980, můžeme některé myšlenky označit jako nadčasové[4]. Podle Paperta mají někteří žáci model učení postavený na schématu, ve kterém je výsledek špatně nebo správně, neexistuje jiná možnost. Ale v programování většinou prvotní verze programu není správně, je potřeba najít a opravit chyby. V tomto schématu pak není hlavní otázkou zda je program správně nebo špatně, ale zda je opravitelný. Další výhodou je, že při programování žáci využívají a učí se matematickému jazyku a matematice, ke kterému mají lepší vztah, protože je pro ně matematika využitelný nástroj, ne cíl výuky.

2.6 Jak učit programování?

Zaměřme se nyní na to, jakým způsobem je možné programování vyučovat. "Výuka algoritmizace a programování prochází v současné době velkými změnami, které se snaží reagovat na dynamický rozvoj softwarového průmyslu. Dříve využívané metodické postupy, modely vývoje či programovací jazyky nedostačují aktuálním potřebám. (Fojtík 2013)

Většinou probíhá výuka nejprve pomocí procedurálního paradigmatu, během kterého jsou studenti obeznámeni s datovými typy, vytvářením proměnných, operátory, cykly, podmínky apod. Výuka je úzce spojena se syntaxí jazyka, bez které by se student neobešel. Až následně se přechází k výuce objektového programování a pojmům jako třída objekt dědičnost. Jednou z nevýhod takového přístupu je, že: „Žáci z tohoto postupu získají pocit, že objektové programování je jen určitá nadstavba jazyka (Fojtík 2013)." Výuka může probíhat i pomocí jiných metodik, představme si krátce některé z nich:

- **Object first** metodika klade důraz na důležité koncepty objektově orientovaného programování (BlueJ, n. d.), aniž by se studenti museli zabývat syntaxí programovacího jazyka. Autoři této metodiky vytvořili vývojové prostředí BlueJ pro objektově orientovaný programovací jazyk Java. Toto prostředí je jednoduché na ovládání a dokáže vizualizovat třídy programu do diagramu.

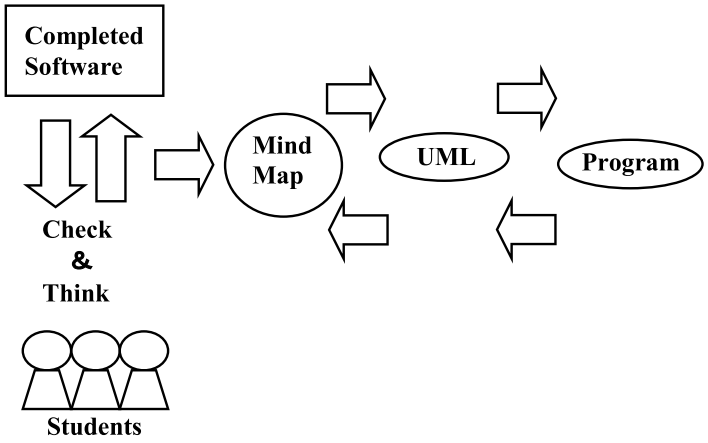
- **Architecture first** metodika se snaží rozšiřovat pojetí Object first metodiky, ke které přidává ještě větší důslednost na znalost architektury softwaru, než se přejde k samotnému kódování programu (Pecinovský 2013).

- **Algorithm first** metodika se snaží zaměřit více pozornosti na vytváření algoritmů, než na jejich následném kódování. Studenti tráví více času navrhováním algoritmu a jejich vizualizaci pomocí vývojových diagramů. Díky tomu by měli být schopni převést problémy reálného světa do podoby algoritmu (Oroma et al. 2012).

- **Agilní metodiky** jsou skupiny metod dodržující k vývoji programů několik základních principů například že "nejúčinnějším a nejefektnějším způsobem

[4] Pepert úzce spolupracoval i s Jeanem Piagetem, tvůrcem známe teorie kognitivního vývoje, která se na pedagogických fakultách učí dodnes

sdělování informací vývojovému týmu z vnějšku i uvnitř něj je osobní konverzace“ nebo “hlavním měřítkem pokroku je fungující software“ (Beck et al. 2001). Studenti v tomto přístupu zastávají roli programátorů, vzájemně diskutují a pracují na návrhu softwaru, ke kterému požívají myšlenkové mapy a následně UML diagramy. Poté se přistupuje k samotnému programování, při kterém se snaží napsat alespoň malou ale hlavně funkční část kódu, což by je mělo motivovat do další práce. Případné problémy mohou vyřešit v kooperaci s ostatními (Kofune – Koita 2012).



Obrázek 2.4: Schéma pracovního postupu při použití agilní metodiky

- **Game first** Tato metodika si klade jako prioritu zaujmout studenty. Výuka probíhá formou vývoje počítačové hry, při kterém by se měli studenti seznámit i s odbornými znalostmi programování. (Leutenegger – Edgington 2007)

Důležitým aspektem ve výuce programováním je nejen volba paradigmatu a metodiky, ale i vhodného programovacího jazyka. Parker et al. (2006) definovali několik vlastností, které by měl mít programovací jazyk vhodný pro vzdělávání jako například:

- jednoduchost použití
- jednoduchá syntaxe
- dobré testovací nástroje
- smysluplné názvy klíčových slov
- použití v komerční sféře
- dostupnost knih
- nezávislost na operačním systému atd.

Uveďme si nyní několik programovacích jazyků, které jsou v současnosti relevantní ve výuce programování. Jedná se o jazyky, které jsou v českém prostředí známé (články o nich se objevují ve sbornících odborných konferencí) a jsou vhodné pro výuku programování jak uvádí ve svém výzkumu Manilla (2006):

- Jazyk **Python** byl poprvé vydán v roce 1991, jeho autorem je Guido Van Rossum. Tento jazyk je interpretovaný a multiparadigmatický, obsahuje konstrukce objektově orientovaného, funkcionálního i imperativního paradigmatu.

Jeho výhodou je přehlednost, byl vytvořen s důrazem na výuku programování. Využívá se k výuce na předních amerických univerzitách (Guo 2014), může být použit i k výuce na základních školách. Podle Hájka a Petera (2015) se „jazyk sa rýchlo a dobre učí. Programovanie je pragmatické a rýchlo vedie k stručnému a efektívnemu programovému kódu.“ (Hájek – Peter 2015)

- **Java** je objektově orientovaný programovací jazyk, vyvinutý v roce 1990 Jamesem Gasolinem ve firmě Sun Microsystem. Jeho syntaxe vychází z populárních jazyků C a C++ (Gosling et al. 2014). Hojně se využívá i k výuce programování na VŠ. Výhodou je značná nezávislost na zařízení, je tak využívána např. na mobilních zařízeních jako mobilní telefony nebo tablety. Pro Javu vzniklo několik vývojových prostředí vhodných pro výuku programování, které umožňuj vizualizovat jednotlivé třídy a objekty programu jako např BlueJ (Pecinovský 2004) nebo Greenfoot (Greenfoot 2016).

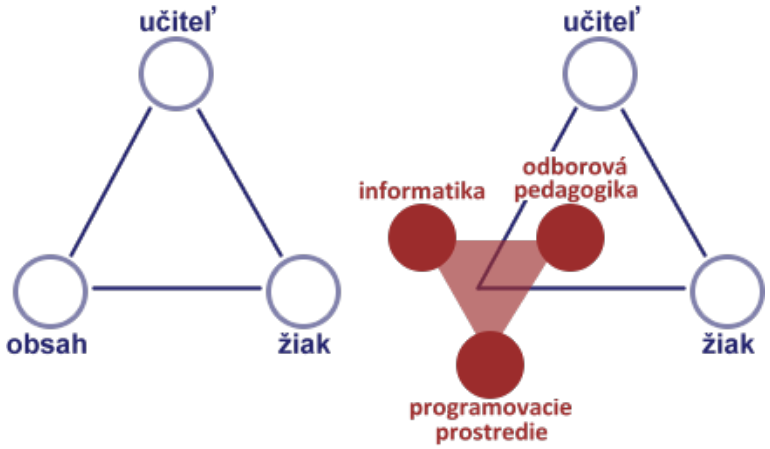
- Programovací jazyk **Scratch** je speciálně vytvořen pro výuku dětí. Patří mezi vizuální programovací jazyky, programování tak probíhá manipulací s grafickými prvky. I přes svoji jednoduchost dovoluje vytvořit i složitější programátorské konstrukce. Další jeho výhodou je možnost jednoduše sdílet vytvořené projekty v komunitě. Ačkoliv se nejedná o klasický, komerčně používaný jazyk, je vhodný k představení základů a vizualizaci konceptů programování.(Musilek 2013)

- Jazyk **Scheme** vznikl v roce 1975, jeho autoři jsou Guy Lewis Steele a Gerald Jay Sussman. Vychází z programovacího jazyka Lisp, je ale značně zjednodušen, „je možné syntaxi jazyka Scheme úplně vyložit během jedné přednášky (Skoupil 2007a, s. 11)“. Jedná se o multiparadigmatický jazyk, „jeho podstatou je programovací paradigma funkcionální, je v něm ale možné pracovat imperativním nebo objektově orientovaným stylem. Scheme je označován také jako algoritmický jazyk, protože je v něm možné snadno formulovat algoritmy. (Skoupil 2007a, s. 10)“

V současnosti může výuka programování probíhat za pomoci mnoha různých programovacích jazyků, vývojových prostředí a dalších nástrojů např. pro sdílení hotových programů. Každý jazyk, prostředí i nástroj však může mít jiný dopad na rozvoj gramotnosti člověka, jak zmiňují Proctor a Blikstein (Proctor – Blikstein 2016). Ti vytvořili tři kategorie, do kterých je jednotlivé jazyky a nástroje možné rozdělit (zestručněno):

- materiální – soustřeďují hlavně na výuku práce s kódem,
- kognitivní – napomáhají rozvoji rekurzivního a informatického myšlení a abstrakce,
- sociální – dovolují využívat kód vytvořený někým dalším nebo se zapojovat do různých komunit.

Je důležité, aby student učitelství informatiky uvědomoval široké možnosti, jak na své žáky pomocí programování působit a pokud možno udržoval jednotlivé aspekty v rovnováze, což zmiňuje ve své práci i Kalaš (2016). Ten vytvořil rozšířenou verzi didaktického trojúhelníku, ve které rozšiřuje vrchol obsahu na tři podvrcholy – informatika, oborová didaktika a programovací prostředí. Tyto tři složky by měly být ve výuce stejně zastoupeny, jinak by mohlo podle autora dojít k deformacím. Pokud se obsah zaměřuje hlavně na programovací prostředí, může to vést k „technocentrizmu“, pokud převládá informatická složka, Kalaš ji doslova označuje jako „Computer Sience centrizmus“, což popisuje jako přehnanou snahu zpopularizovat obor informatika na úkor zbylých dvou složek. Nakonec preference didaktického obsahu může vést k “plytkému nebo žádnému porozumění programovacích konstruktů“.



Obrázek 2.5: Rozšířený didaktický trojúhelník

Pokud bereme v potaz vše výše uvedené, je zřejmé, že teorie výuky programování dosahuje značné šíře. Aby si byl budoucí učitel informatiky mohl být vědom možností a potencionálních problémů, které výuka programování může představovat, je tak důležité, aby studenti byly obeznámeni s didaktikou programování a to nejlépe v samostatném předmětu. Jen tak budou moci studenti chápat i další souvislosti výuky programování a chápat jeho pozici ve výuce, protože například jak píše Lessner (Lessner 2013, s. 13):

> „Programování jako takové (vývoj softwaru, popř. zápis algoritmů) ale na gymnáziu těžko obstojí v roli vzdělávacího cíle. Tímto cílem může být kultivace myšlení, rozvoj schopnosti systematicky analyzovat a řešit problémy, nikoliv znalost konkrétního programovacího jazyka.“

Proto by i výuka programování měla brát v první řadě důraz na pochopení algoritmů před důkladnou znalostí syntaxe jazyka.

3 Analýza deklarovaného kurikula pregraduální přípravy

Příprava budoucích učitelů informatiky probíhá v ČR na přírodovědědeckých a pedagogických fakultách v rámci strukturovaných dvoustupňových programů. V praktické části této práce byl nejdříve proveden výzkum skládající se z rešerše všech dostupných bakalářských i magisterských programů pro pregraduální přípravu učitelů informatiky. V programech byly vybrány předměty související s problematikou programování a algoritmizace. Samotným výzkumem je obsahová analýza těchto předmětů, na základě které bylo provedeno porovnání jednotlivých programů. V každé části výzkumu je popsána příslušná metodika.

3.1 Stav problematiky

V roce 2013 analyzoval programy pro pregraduální přípravu učitelů informatiky Berki (2013). Bylo provedeno porovnání akreditovaných programů hlavně z hlediska poměru počtu kreditů pro předem definované kategorie (matematika, algoritmy, databázové a operační systémy, publikační systémy, technologie počítačů, didaktika). Podle této studie bakalářské programy zaměřují spíše na odbornou informatiku, zatímco navazující magisterské programy se zaměřují spíše na didaktiku informatiky. Výsledky dále ukázaly na velkou kreditovou dotaci v oblasti algoritmů (do které byla zařazena výuka programovacích jazyků, formálních jazyků, gramatik a automatů), kdy tato kategorie na většině fakult zaujímala více než čtvrtinu kreditů v bakalářských programech.

3.2 Popis zkoumaného vzorku

Předmětem zkoumání byly studijní programy zaměřené na pregraduální přípravu učitelů informatiky základních a středních škol. Studie se zaměřuje na prezenční studium jakožto nejčastější formu studia, nejsou jde zahrnutý kombinované ani dálkové programy. Pro relevantnost studie byly do výzkumu zahrnuty pouze programy, do kterých jsou přijímáni studenti pro rok 2017/2018. Tímto opatřením bylo zajištěno, že budou zahrnuty pouze programy s platnou akreditací.

V ČR najdeme i některé programy, které spojují informatiku a technickou výchovu a jsou určeny hlavně pro budoucí učitele odborných středních škol nebo učitele

technické výchovy na ZŠ. Tyto programy se skladbou předmětů značně liší, např. obsahují více odborně zaměřených předmětů zaměřených na výuku programování nebo teorii operačních systémů, pro svoji specifičnost tak do výzkumu zařazeny nebyly.

Jelikož mohou VŠ akreditovat programy vzdělávající budoucí učitele informatiky pod různými jmény, byla provedena rešerše oborů na všech fakultách vzdělávajících učitele v ČR. Z této rešerše vznikl vzorek 25 studijních programů ze 14 fakult 10 univerzit:

- Jihočeská univerzita v Českých Budějovicích (JU) – Pedagogická fakulta (PedF), Přírodovědecká fakulta (PřF);
- Masarykova univerzita (MU) – Fakulta informatiky (FI);
- Ostravská univerzita – Pedagogická fakulta (PedF), Přírodovědecká fakulta (PřF);
- Technická univerzita v Liberci (TUL) – Fakulta přírodovědně-humanitní a pedagogická (FP);
- Univerzita Hradec Králové (UHK) – Přírodovědecká fakulta (PřF), Pedagogická fakulta (PedF);
- Univerzita Jana Evangelisty Purkyně v Ústí nad Labem (UJEP) – Přírodovědecká fakulta (PřF);
- Univerzita Karlova (UK) – Matematicko-fyzikální fakulta (MFF), Pedagogická fakulta (PedF);
- Univerzita Palackého v Olomouci (UP) – Přírodovědecká fakulta (PřF);
- Univerzita Tomáše Bati ve Zlíně (UTB) – Fakulta aplikované informatiky (FAI);
- Západočeská univerzita v Plzni (ZČU) – Fakulta pedagogická (PedF).

Většina fakult nabízí pro učitelství jeden bakalářský a jeden magisterský program, vyjímkou jsou PřF UJEP a PedF OU které nabízí pouze bakalářský program a na FAI UTB najdeme pouze navazující magisterský program. FP TUL nabízí v rámci NMgr studia 2 programy – učitelství pro 2. stupeň i pro SŠ. Specifická je situace na UHK, kde výuku bakalářů zajišťuje PřF a navazující magisterské studio přebírá PedF opět ve dvou programech pro ZŠ i SŠ.

Pro potřeby studie bylo nutné vybrat předměty, které souvisí s výukou programování. V prvním kroku byly analyzovány názvy předmětů v získaných studijních programech. Na základě této analýzy a poznatků z teoretické části práce sestavena množina klíčových slov a sousloví, které by se mohly objevit v názvech předmětů, jež souvisí s tématem programování a algoritmizace nebo příbuzných tématech. Tato množina obsahovala tyto klíčová slova:

> *programování*, *programovací jazyk*, *paradigma*, jména různých programovacích jazyků jako *Java*, *C++* (jazyky byly průběžně doplňovány podle

zjištěných výsledků, nelze tuto skupinu definovat předem), dále slova související s teorií algoritmu – *algoritmus*, *datové struktury*, *složitost*, *vyčíslitelnost*, klíčová slova, která by se mohla objevit v názvu souvisejících s didaktikou programování a robotikou – *didaktika*, *robotika* nebo slova, která by mohla souviset s předměty obsahující programovací jazyky určené pro vývoj webových aplikací –*www*, *web*.

Do prvotní množiny předmětů byly zařazeny všechny ty, jejichž název obsahoval některé z výše zmíněných klíčových slov. Dále byly přidány ty, u kterých z jejich názvů nebylo jasné jejich zaměření, a přesto mohly s algoritmizací nebo programováním souviset, např. předmět s názvem *Vstupně výstupní komunikace*.

Do studie byly zařazeny výhradně povinné předměty a skupiny povinně volitelných předmětů, ve kterých jsou všechny předměty stejného zaměření, např. skupina programovacích jazyků, ze kterých si student musí jeden vybrat. Aby nebyly výsledky zkreslené, je celá skupina do výsledků 1. fáze výzkumu zastoupena jako jeden předmět.

Obsahy sylabů všech předmětů z této prvotní skupiny byly dále podrobněji zkoumány, a rozděleny do čtyř pojmenovaných skupin. Tyto skupiny se dělí podle výskytu klíčových slov ve dvou částech jednotlivých sylabů – obsahu předmětu a jeho cílech.

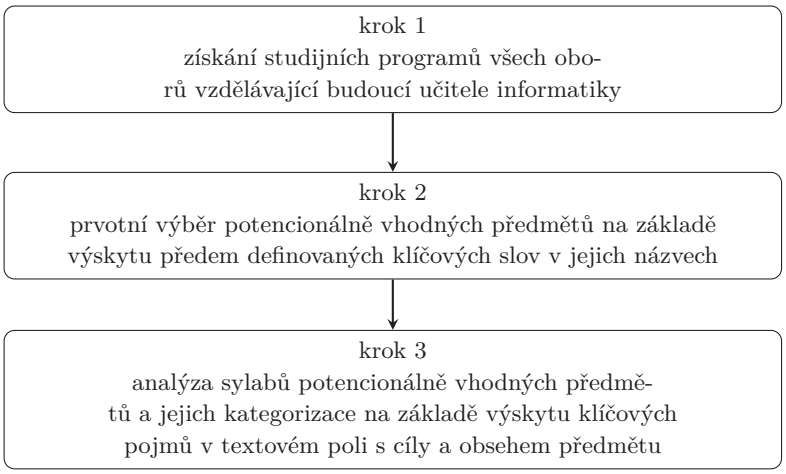
- **Výuka programování (VP)** – do této kategorie byly zařazeny všechny předměty, jejichž obsahem je výuka konkrétního programovacího jazyka nebo je některý z jazyků použit při výuce. Dále jsou zde zařazeny předměty jejichž teorie přímo související s programováním, jako je teorie paradigmat programování. V sylabech těchto předmětů najdeme klíčové pojmy: *programování*, *programovací jazyk*, *programovací paradigma*, názvy konkrétních programovacích paradigmat, jako *objektově orientované*, *funkcionální*, názvy konkrétních programovacích jazyků jako *Java*, *C++*.
- **Výuka algoritmizace (VA)** – do této kategorie jsou zařazeny všechny předměty, jež se zaměřují na výuku teorie algoritmů, datových struktur a teorie složitosti a vyčíslitelnosti algoritmů. Klíčové pojmy: *algoritmizace*, *algoritmus*, *datové struktury*, *složitost*, *vyčíslitelnost*.
- **Didaktické předměty (DP)** – do této kategorie jsou zařazeny předměty, které se zaměřují na didaktiku programování nebo didaktiku algoritmizace, či jsou tato témata obsažena v rámci didaktiky informatiky. Dále jsou zde zařazeny předměty, v nichž je zařazena výuka dětských programovacích jazyků a robotiky[1]. Klíčové pojmy: *didaktika programování*, *didaktika algoritmizace*, *dětský programovací jazyk*, *robotika*. Jelikož témata v sylabu mohou mít stejné názvy jako témata ze skupin VP a VA, bude zohledněn i název předmětu.
- **Programování v prostředí WWW (WWW)** – do této kategorie jsou zařazeny všechny předměty, v nichž se objevuje programování webových aplikací či stránek v některém z programovacích jazyků pro tento účel vhodných.

[1] předpokládá se, že robotika v učitelských programech slouží jako vyučovací nástroj pro výuku programování a algoritmizace

V sylabech těchto předmětů najdeme klíčové pojmy: *PHP*, *JavaScript*, *webová aplikace*.

Pro zařazení předmětu není dána žádná minimální hranice příslušných pojmů, stačí jediná zmínka vztahující se k jedné ze skupin, aby tento předmět do ni byl zařazen (pokud nejsou přítomna témata z jiných skupin.

V případě, že předmět obsahuje klíčové pojmy z více skupin, je zařazen do té, která pokrývá větší část jeho obsahu (je této skupině v předmětu věnováno více témat).



Obrázek 3.1: Metodika 1. fáze výzkumu

3.3 Výsledky 1. fáze výzkumu

Metodikou popsanou výše bylo získáno a kategorizováno celkem 61 předmětů v bakalářských programech studia a 34 předmětů v navazujících magisterských programech. Data jsou zde prezentována v tabulkách, šedivou barvou jsou označeny položky s nulovou hodnotou. V průběhu rozřazování bylo zjištěno, že programování a algoritmizace jsou v některých předmětech zastoupeny skoro v rovné míře, což naznačují i poznatky z teoretické části práce. Výuka programování je vlastně "zhmotnění" teorie algoritmů do podoby hotových programů a naopak algoritmy mohou být představeny pomocí programových konstrukcí.

3.3.1 Bakalářské programy

Podle výsledků v tabulce 4.4 jsou bakalářské programy zaměřeny hlavně na výuku programování (kategorie VP), na kterou v průměru vymezují 2,33 předmětů v průběhu studia. Výuku programování najdeme na všech zkoumaných studijních programech. Jako druhou nejvíce zastoupenou se ukázala výuka algoritmizace zastoupena v průměru s 1,75 předměty na studium. Výuka probíhá na všech univerzitách kromě PedF UK.

Tabulka 3.1: Počet předmětů předmětů v bakalářském studiu

	PedF JU	PřF JU	FI MU	PřF OU	PedF OU	FP TUL	PřF UHK	PřF UJEP	MFF UK	PedF UK	PřF UP	PedF ZČU	*prům.*	*med.*
VP	1	2	3[1]	1	2	3	2	2	4[2]	2	4	2	2,33	2
VA	1	2	3	2	1	1	2	2	2	0	3	2	1,75	2
DP	2	0	0	0	0	0	1	0	0	1	0	0	0,33	0
WWW	0	1	0	0	1	0	1	0	0	1	0	1	0,42	0
celkem	4	5	6	3	4	4	6	4	6	4	7	5	4,83	4

[1] možnost výběru ze dvou programovacích jazyků v jednom semestru
[2] možnost výběru ze tří programovacích jazyků v jednom semestru

Výuka didaktiky programování nebo algoritmizace není v bakalářských studijních programech většinou zastoupena, čímž tento výzkum potvrzuje závěr Berkiho, že bakalářské programy se zaměřují převážně na odborné předměty. Výjimkou je zde výuka na PedF JU, která jako jediná obsahuje dva předměty z kategorie DP.

Výuku programovacích jazyků zaměřených na prostředí WWW většina studijních programů neobsahuje nebo je zastoupena v jednom předmětu.

Celkově mají bakalářské studijní programy v průměru 4,83 předmětů (medián 4), které spadají do některé z kategorií. Rozdíly mezi programy jsou znatelné – zatímco na PřF OU vyhovují kritériu 3 předměty, na PřF UP je to 7 předmětů. Ještě větší rozdíl najdeme, pokud se zaměříme pouze na kategorie VP a VA[2]. V tomto případě najdeme na PedF JU a PedF UK pouze 2 takto zaměřené předměty, kdežto na PřF UP je těchto předmětů 7, případně 6 na FI MU a MFF UK. Dále je potřeba zmínit že na FI MU a MFF UK je možnost zvolit si vystudovaný programovací jazyk[3]. Ze získaných dat vyplývá, že dotace hodin v zkoumané oblasti je pro bakalářské studijní programy rozdílná a to především v předmětech zaměřujících se na výuku algoritmizace a programování.

3.3.2 Navazující magisterské programy

V navazujících magisterských programech bylo kategorizováno celkem 34 předmětů. Nejvíce jsou zde zastoupeny předměty výuky didaktiky programování a algoritmizace (kategorie DP) s průměrnou dotací 1,46 předmětu na studium. Kromě obou programů PeF UHK je předmět kategorie DP zastoupen na všech fakultách. Nejvíce předmětů (3) z kategorie DP je obsaženo v programu PřF JU.

Druhou nejvíce zastoupenou skupinou jsou předměty na výuku algoritmizace s průměrným zařazením 0,46 předmětu na program, kdy většina fakult ani výuku algoritmizace nezařazuje.

[2] Dává smysl tyto dvě kategorie analyzovat souhrně jako jednu, hranice mezi nimi je úzká a některé předměty stojí na jejich pomezí – výuka algoritmizace a programování je zastoupena skoro ve stejné míře.

[3] To je dáno především velikosti fakult a jejich zaměření i na neučitelské programy studia informatiky.

Výuku programování v navazujícím studiu najdeme jen na FAI UTB a obou programech FP TUL s průměrným zastoupením 0.38 předmětu na program.

V průměru nejméně zastoupenou je výuka programování v prostředí WWW s průměrným výskytem 0,31 předmětu na studijní program. Ve 4 studijních programech je do této skupiny zařazen 1 předmět, v ostatních toto téma do výuky zařazeno není.

Celkově kritériím některé z kategorií vyhovovalo 2,62 předmětu na studijní program (medián 3). Při celkovém pohledu na získaná data zjistíme, že v navazujících studijní programy se zaměřují hlavně na výuku didaktických předmětů, zatímco jsou ostatní kategorie obsaženy pouze v malém počtu. Najdeme zde programy, ve kterých je zastoupena výuka všech kategorií – FP TUL ZŠ, i program, který neobsahuje ani jeden předmět vyhovující zadaným kriteriím – PedF UHK ZŠ.

Tabulka 3.2: Počet předmětů předmětů v navazujícím studiu

	PedF JU	PřF JU	FI MU	PřF OU	FP TUL ZŠ	FP TUL SŠ	PedF UHK ZŠ	PedF UHK SŠ	MFF UK	PedF UK	PřF UP	FAI UTB	PedF ZČU	*prům.*	*med.*
VP	0	0	0	0	2	2	0	0	0	0	0	1	0	0,38	0
VA	1	0	0	0	1	0	0	1	1	0	2	0	0	0,46	0
DP	2	3	2	1	1	2	0	0	2	1	1	2	2	1,46	2
WWW	0	0	0	0	1	1	0	0	1	0	0	1	0	0,31	0
celkem	3	3	2	1	5	5	0	1	4	1	3	4	2	2,62	3

3.3.3 Celkový pohled

Ačkoliv je v ČR zaveden model strukturovaného studia, můžeme porovnat výuku na jednotlivých fakultách během celého 5letého cyklu. V tabulce 3.3 najdeme počet předmětů v jednotlivých kategoriích, které studenti absolvují během součtu 3letého bakalářského a 2letého navazujícího magisterského studia na jedné fakultě[4], fakulty jež neobsahují Bc. i NMgr. program v tomto přehledu uvedeny nejsou.

Nejvíce zastoupenou skupinou předmětů jsou předměty spadající do kategorie výuky programování s průměrným počtem 2,75 předmětů na studium (medián 2). Programování zařazují do svých programů všechny fakulty a to v širokém rozpětí 1-5 předmětů. Nejvíce předmětů spadajících do této kategorie najdeme v obou programech TUL a to v počtu 5 předmětů. Naopak nejméně předmětů z této kategorie je zařazeno v programech PedF JU a PřF OU.

Druhou nejvíce zastoupenou skupinou předmětů je výuka algoritmizace (VA) s průměrným obsahem 2,33 předmětu na studium (medián 2). Kromě PedF UK najdeme předmět z této kategorie ve všech studijních plánech a to v rozsahu 0–5 předmětu na studium, kdy nejčastěji je zařazen v počtu 2 předmětů. Dalším zjištěním je podobný počet předmětů VA a VP pro každou z fakult, kromě FP TUL a

[4] pro UHK zaštiťuje bakalářský program přírodovědecká fakulta, uveden je proto součet PřF a PedF UHK

Tabulka 3.3: Počet předmětů předmětů v součtu

	Ped-FJU	PřF JU	FI MU	PřF OU	FP TUL ZŠ	FP TUL SŠ	PedF UHK ZŠ	PedF UHK SŠ	MFF UK	PedF UK	PřF UP	PedF ZČU	*prům.*	*med.*
VP	1	2	3	1	5	5	2	2	4	2	4	2	2,75	2
VA	2	2	3	2	2	1	2	3	3	0	5	2	2,25	2
DP	4	3	2	1	1	2	1	1	2	2	1	2	1,83	2
WWW	0	1	0	0	1	1	1	1	1	1	0	1	0,67	1
celkem	7	8	8	4	9	9	6	7	10	5	10	7	7,5	7,5

PedF UK je rozdíl v počtu předmětů maximálně jeden předmět. Můžeme tak vyslovit tvrzení, že fakulty ve svých programech mají tendenci nabízet podobný počet předmětů na výuku algoritmizace jako programování.

Třetí nejvíce zastoupenou kategorií je DP s průměrným výskytem 1,83 předmětu na studium (medián 2). Pro tuto kategorii jsou pro většinu univerzit vymezeny 1–2 předměty, výjimkou jsou pouze obě fakulty JU – Přírodovědecká fakulta se 3 didaktickými předměty a Pedagogická fakulta se 4 předměty.

Nejméně zastoupenou skupinou jsou předměty na výuku programování v prostředí WWW s průměrem 0,67 předmětu na studium. V této kategorii se fakulty shodují, rozsah výskytu předmětu z této kategorie je pouze 0–1.

V celkovém součtu je součástí studijních programů v průměru 7,5 kategorizovaných předmětů. Liší se mezi sebou především v zastoupení předmětů na výuku programování a algoritmizace, v ostatních kategoriích je zastoupení předmětů podobné. Právě kvůli rozdílů, v zastoupení v kategoriích VP a VA jsou i rozdíly mezi fakultami v celkovém počtu kategorizovaných předmětů značné a to v rozsahu 4–10 předmětu na studium. Nejméně kategorizovaných předmětů nabízí PřF OU v počtu 4, naopak nejvíce předmětů nabízí MFF UK a PřF UP v počtu 10 předmětů na studium.

Podle získaných dat je tak studium učitelství informatiky nejednotné v zastoupení výuky algoritmizace a programování, což může mít za následek i rozdíl v kvalitě vzdělání absolventů v této oblasti. Jak bylo zmíněno už v kapitole 2, součástí výuky učitelů informatiky by měla být i didaktika programování a algoritmizace, která podle získaných výsledků je ale ve studiu zastoupena v malém počtu předmětů[5]. Bude předmětem dalšího zkoumání, v jaké formě se didaktika programování a algoritmizace objevuje – zda v samostatných předmětech, nebo jako součást didaktiky informatiky, a na jaká témata se zaměřuje.

Na kvalitu vzdělání nemá vliv jen množství absolvovaných předmětů ale i jejich rozmístění do průběhu celého studia. Obsah jednotlivých předmětů by na sebe měl navazovat, později absolvované předměty by měli syntetizovat poznatky z těch dříve absolvovaných, případně prohlubovat úroveň dosažených znalostí. Studijní programy mají ve většině případů pro jednotlivé předměty vymezen semestr, ve kterém by

[5] Nehledě na to, že jsou do výzkumu zařazeny i předměty, ve které didaktika programování zaujíma jen část výuky.

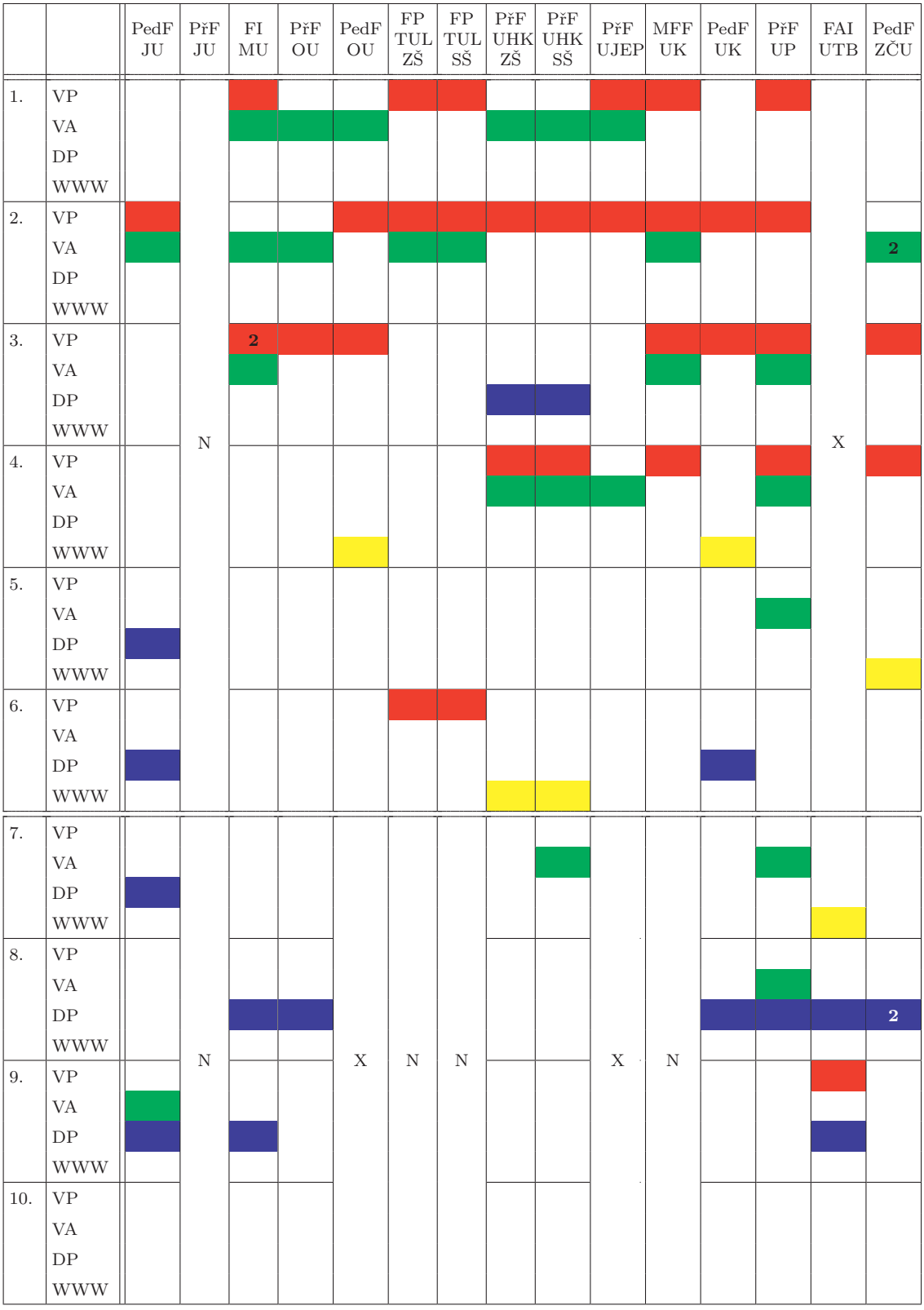
měly být absolvovány, z těchto údajů byla sestavena tabulka 3.4. Řádky tabulky představují semestry studia, sloupce jednotlivé studijní programy. Výskyt předmětu spadající do některé ze zkoumaných kategorií je označen obdélníkem, pro přehlednost jsou jednotlivé kategorie barevně odlišené. Pokud se v jednom semestru vyučuje více předmětů z jedné kategorie, číslo v obdélníku označuje počet těchto předmětů.

Podle zjištěných dat fakulty ve svých studijních programech nasazují většinu předmětů spadajících do jedné z kategorií v prvních čtyřech semestrech bakalářského studia. Většina programů obsahuje už v prvním semestru předmět zaměřený na výuku programování nebo algoritmizace, s výukou těchto předmětů se začíná nejdéle ve druhém semestru. Pokud budeme zjišťovat s kterou kategorií předmětů se studenti setkají jako první, většina fakult (PřF OU, PedF OU, PřF UHK, MFF UK, PedF ZČÚ) volí jako první předmět zaměřený na výuku algoritmizace, FP TUL, PedF UK a PřF UP jako první vyučují předmět zaměřený na programování a PedF JU, FI MU a PřF UJEP vyučují předměty z obou skupin v jednom semestru.

Didaktické předměty jsou ve většině případů zařazeny do navazujícího studia, a tedy se vyučují se ze všech kategorizovaných předmětů jako poslední. Výjimkou je UHK která ve svém bakalářském programu zařazuje didaktický předmět už do třetího semestru bakalářského studia.

Předměty zaměřující se na výuku programování v prostředí WWW jsou umístěné hlavně do druhé poloviny bakalářského studia, studenti v nich stejně jako v didaktických předmětech mohou využít znalosti nabyté v předchozím studiu algoritmizace a programování. Je třeba upozornit na nedostate použité metodiky – do výzkumu jsou zařazeny i předměty, jež nejsou celé věnovány zkoumaným oblastem a sylabus se o těchto tématech jen letmo zmiňuje. V další fázi výzkumu bude míra zastoupení klíčových témat v předmětu konkretizována.

Tabulka 3.4: Rozložení předmětů do průběhu studia



* X fakulta nemá pro tyto semestry program
N studijní plán neobsahuje doporučený průběh studia pro všechny předměty

4 Textová analýza

Základní výzkumnou metodou se stala obsahová analýza, která byla zvolena jako relevantní způsob pro získání užitečných dat. Obsahová analýza je výzkumný nástroj k určení přítomnosti určitých slov nebo konceptů v textu či skupině textů (Writing@CSU 2017, Dvořáková 2010). Pomoci této metody můžeme objektivně, systematicky a kvantitativně popsat obsah nějakého druhu lidské komunikace (Švec 1998), v našem případě jsou to sylaby předmětů, které byly vybrány podle určitých pravidel popsaných v kapitole 3.2. Dvořáková (2010) popisuje dva druhy obsahové analýzy – konceptuální a relační. Konceptuální se zaměřuje na kvantifikaci přítomnosti určitého znaku, kdežto relační bere v úvahu i vztahy mezi jednotlivými znaky. V tomto výzkumu byla zvolena konceptuální analýza, která se více hodí pro komparaci více datových souborů.

Obsahová analýza se skládá z různých komponentů, které je možné přizpůsobit podle potřeb výzkumu. Gavora uvádí přehled těchto komponentů s možnostmi jejich konfigurace (Gavora 2015):

- typ obsahu – mediální, bez omezení;
- výběr vzorku – stochastický (náhodný), intencionální (záměrný);
- hloubka analýzy – manifestní (zaměřující se jen na zjevný obsah v textu), latentní (hledá kontext i „mezi řádky“);
- směr postupu – deduktivní (od teorie k datům), induktivní (od dat k závěrům);
- druh kontroly dat – intersubjektová shoda (shoda na závěrech výzkumu mezi výzkumníky), kredibilita (věrohodnost);
- forma výsledků výzkumu – numerický, verbální.

Gavora však upozorňuje na nemožnost konfigurovat jednotlivé komponenty libovolně, protože by některé konfigurace neměly smysl. V souladu s tímto textem byla zvolena koncepce s intencionálním (záměrným) výběrem vzorku, manifestní analýzou, induktivním směrem postupu, kreditabilním druhem kontroly dat a verbální i numerickou formou prezentace výzkumu. Dále uvádí i posloupnost kroků, jež by se měla držet obsahová analýza s induktivním směrem postupu:

1. stanovení výchozího metodického přístupu, vysvětlení hlavního hlediska, ze kterého budeme nahlížet na obsah, který budeme analyzovat;
2. stanovení předběžných výzkumných otázek (foreshadowed questions);

3. stanovení jednotky analýzy;
4. stanovení vzorku textů;
5. analýza dat, stanovení obsahových kategorií induktivním způsobem, jejich definovaní, exemplifikace a zpětné ověření (tento postup se opakuje, dokud se nenajde nejlepší řešení – neodstraní se nejednoznačnosti, protiřečení, překrývání kategorií apod.);
6. seskupování a uspořádání kategorií podle smyslu (podle potřeby se kategorie organizují do několika rovin s odstupňovanou úrovní abstraktnosti);
7. reflexe a kritická prověrka předchozího postupu výzkumníkem;
8. napsaní deskriptivního a interpretačního souhrnu, včetně použití autentických citátů z analyzovaného materiálu, přitom se využívají různé modely, matice a schémata.

4.1 Popis výzkumu a jeho metodického přístupu

Cílem výzkumu je porovnat studijní programy zaměřující se na výuku budoucích učitelů informatiky a to speciálně v oblasti programování a algoritmizace. Studijní programy bylyv první fázi výzkumu podrobeny filtraci, jejímž účelem bylo vybrat ze všech předmětů ty, které mají souvislost s výukou programování a následně je kategorizovat. Díky kategorizaci, sestavení tabulky četností těchto předmětů a sestavení přehledu jejich časového zasazení do průběhu studia bylo sice možné vytvořit letmé porovnání studijních programů napříč republikou, ale je potřeba toto porovnání dále rozvést a zaměřit se i na obsah jednotlivých předmětů.

VŠ deklarují obsah svých předmětů v sylabech, které mají pro každou fakultu přibližně stejný rámec – najdeme zde obsah předmětu, jeho cíle, získané způsobilosti a literaturu, liší se ale způsobem, jakým jsou tyto datová pole vyplněny. Například v popisu obsahu předmětu je v některých případech vypsán obsah každé z přednášek a cvičení, někdy pouze tématické celky, na které je předmět zaměřen. Přímá komparace sylabů jednotlivých předmětu jako celků mezi sebou by nebyla možná (nelze například s jistotou určit, kolik času věnuje přednášející určitému tématu).

Budeme tak nahlížet na sylaby jako na dokumenty, které obsahují především výčet jednotlivých témat, která jsou v předmětu obsažena (která jsou v předmětu vyučována, případně slouží jako výuková metoda) a jejíž existenci v dokumentu lze potvrdit či vyvrátit. Tato témata můžeme v sylabech hledat v podobě jednoslovných pojmů (např. názvy programovacích jazyků jako *Java*, *C++*) či víceslovných pojmů (např. *objektově orientované programování*). Skladba pojmů, na které budeme jednotlivé sylaby testovat vychází především z teoretické části této práce, kde byla nejdůležitější témata definována[1] a z předběžných výzkumných otázek, které je nutné si před začátkem práce určit.

[1] Jedná se o názvy programovacích paradigmat, programovacích jazyků, konceptů výuky programování, či testování na pojmy související s didaktikou algoritmizace a programování.

4.2 Stanovení výzkumných otázek

Před provedením samotné analýzy je nutné si stanovit, na které otázky by měl tento výzkum odpovídat, podle čehož bude i sestavena množina kódových slov – tedy těch slov, jejichž existenci v textu zjišťujeme. Otázky se zaměřují hlavně na témata, která jsou stěžejní pro výuku, a jež jsou relevantní pro porovnání jednotlivých programů. Otázky můžeme rozdělit do jednotlivých témat:

Programovací jazyk – Jaký programovací jazyk se používá při výuce programování a algoritmizace? S kolika programovacími jazyky se studenti během studia setkají? Existuje mezi studijními programy shoda ve výběru? Jaké programovací paradigma převažuje?

Metodika výuky – Jaké metodiky jsou využívány při výuce programování? Zmiňují se vůbec sylaby o způsobu, jakým je programování vyučováno?

Didaktika programování a algoritmizace – Jsou pro didaktiku programování a algoritmizace vyhrazeny vlastní předměty, nebo jsou tato témata součástí kurzů didaktiky informatiky? Jsou součástí výuky témata jako unplugged teaching či robotika?

4.3 Stanovení jednotky analýzy

Našim cílem je porovnat jednotlivé studijní programy na základě otázek stanovených výše, je nutné ale detailně definovat způsob jakým bude analýza probíhat, aby bylo dosaženo relevantnosti odpovědí. V této analýze se soustředíme na kategorizaci předmětů na základě předem definovaných parametrů (tyto parametry se budou pro každou výzkumnou otázku lišit) základní jednotkou analýzy proto volíme sylabus jednoho předmětu. Tyto jednotky budou na základě určitých pravidel kategorizovány a výsledky následně interpretovány.

4.4 Stanovení vzorky textu

Jelikož provádíme porovnání studijních programů, není potřeba pro kredibilitu používat nějakou formu náhodného výběru. Analyzované jednotky jsou tak sylaby všech předmětů které byly vybrány v první fázi výzkumu jako vztahující se k tématu algoritmizace a programování. Jelikož pro potřeby studie nebylo nutné evidovat a zkoumat sylaby v jejich kompletní podobě (což vlastně nebylo ani do důsledku možné, protože se podoba sylabů napříč republikou liší), byla tak pro každý sylabus evidována tyto pole:

- název předmětu;
- doporučený semestr absolvování předmětuů;
- kreditová dotace předmětu;

- hodinová dotace předmětu;
- cíle předmětu;
- obsah předmětu;
- získané způsobilosti;
- literatura vztahující se k předmětu.

4.5 Výsledky

4.5.1 Použitý programovací jazyk

První kritérium, na které se v našem výzkumu zaměříme, je požitý programovací jazyk. V 1. fázi výzkumu byly získány a označeny ty předměty, které se na výuku programování zaměřují, následně pak bylo možné provést hlubší rozbor pomocí obsahové analýzy. V této části byly zkoumány všechny předměty zaměřující se na výuku programování, tedy všechny ty, které byly v 1. fázi výzkumu zařazeny do kategorie VP. Předměty byly dále kategorizovány podle programovacího jazyka, který se při výuce používá. Předmět byl přiřazen k programovacímu jazyku v případě, že sylabus tohoto předmětu obsahoval název programovacího jazyka ve svém názvu nebo v dalších částech – cílech nebo obsahu. Pokud předmět neobsahoval název programovacího jazyka, byl zařazen do kategorie „neurčeno“. Pokud naopak jeho sylabus zmiňoval více programovacích jazyků, pro každou kombinaci byla vytvořena speciální kategorie. Výsledky analýzy jsou prezentovány formou tabulky – číselné údaje jsou počty předmětů s výukou příslušného programovacího jazyka pro jednotlivé fakulty. Pokud se v jednom předmětu vyučují dva programovací jazyky, v tabulce je tato skutečnost reflektována použitím desetinného čísla – pro každý jazyk připadá 0,5 předmětu. Na FI MU mají studenti možnost si v jednom semestru zvolit z nabídky dvou programovacích jazyků. na MFF UK dokonce ze tří – tyto předměty jsou v tabulce označeny hvězdičkou.

Celkově bylo identifikováno 11 programovacích jazyků, což ukazuje na nejednotnost výuky napříč fakultami. Z výsledků je patrné, že nejčastěji používaným jazykem pro výuku programování je *Java*, která se používá na 6 fakultách, z toho na dvou má formu povinně volitelného předmětu. Dalším používaným jazykem je programovací jazyk *C*, jež je zastoupen na 4 fakultách, z toho na jedné je jako povinně volitelný. Programovacím jazykem, který se vyučuje na třech fakultách je i *Prolog*, pro který ale nejsou vyhrazený vlastní předměty a vyučuje se společně s dalším programovacím jazykem. Zaměření na programovací jazyky *Java* a *C* můžeme popsat jako snahu fakult připravovat studenty i na aplikaci v komerční sféře, jelikož tyto dva jazyky se zde řadí k nejpoužívanějším (TIOBE 2017). 6 fakult má ve svém programu minimálně jeden předmět zaměřený na výuku programování, který explicitně ve svém sylabu nezmiňuje programovací jazyk použitý při výuce (tedy předměty z kategorie neurčeno), na 4 fakultách se žádný ze sylabů nezmiňuje o použitém programovacím jazyku – takto koncipované programy dávají vyučujícím značnou možnost variace přemětu.

Tabulka 4.1: Porovnání výuky programovacích jazyků ve výuce

	PedF JU	PřF JU	FI MU	PřF OU	PedF OU	FP TUL ZŠ	FP TUL SŠ	PřF UHK ZŠ	PřF UHK SŠ	PřF UJEP	MFF UK	PedF UK	PřF UP	FAI UTB	PedF ZČÚ
neurčeno	0	0	0	**1**	0	**1**	**1**	**2**	**2**	**2**	0	**1**	0	0	**2**
C	0	0	**1***	0	0	**1**	**1**	0	0	0	0	0	**2**	0	0
C#	0	0	0	0	0	0	0	0	0	0	**1**+**1***	0	0	0	0
C++	0	0	0	0	0	**0,5**	0	0	0	0	**1***	0	0	0	0
Delphi	0	0	0	0	0	**0,5**	0	0	0	0	0	0	0	0	0
Haskell	0	0	**0,5**	0	0	0	0	0	0	0	**0,5**	0	0	0	0
Java	**1**	**2**	**1***	0	0	**2**	**2**	0	0	0	**1***	**1**	0	0	0
Pascal	0	0	0	0	**1**	0	0	0	0	0	**1**	0	0	0	0
Prolog	0	0	**0,5**	0	0	0	**0,5**	0	0	0	**0,5**	0	0	0	0
Python	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0
Scratch	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0
Scheme	0	0	0	0	0	0	**0,5**	0	0	0	0	0	**2**	0	0

* volitelný předmět

Z tabulky můžeme získat odpověď na další výzkumnou otázku a porovnat, jak se jednotlivé fakulty liší v rozmanitosti nabídky programovacích jazyků v rámci jedno studijního oboru. V tomto ohledu je „nejpestřejší“ výuka na MFF UK, na které se studenti setkají v rámci studia až s 5 programovacími jazyky[2]. Na obou oborech FP TUL i na FI MU se studenti setkají s minimálně 4 programovacími jazyky, na ostatních fakultách už je pak portfolio představených programovacích jazyků znatelně menší, s maximálním počtem dvou programovacích jazyků na průběh celého pětiletého studia. Je důležité zdůraznit, že výsledky nelze pohlížet optikou “čím více, tím lépe“, jedná se pouze o komparativní studii.

4.5.2 Použité programovací paradigma

Jako další parametr výuky programování můžeme označit programovací paradigma. Jaká paradigmata se ve výuce objevují? Shodují se fakulty ve výběru programovacího paradigmatu pro výuku programování? Abychom mohli na tyto otázky odpovědět, byla sestavena tabulka na základě stejné metodologie jako v případě programovacích jazyků, tentokrát ale byla jako klíčová slova použity názvy programovacích paradigmat. Při porovnání s předchozí tabulkou zjistíme, že ačkoliv některé fakulty ve svých sylabech nezmiňují použitý programovací jazyk, je v sylabech definováno použité programovací paradigma, jak je tomu například u ZČU. Dále je nutné upozornit na vztah jednotlivých identifikovaných paradigmat. V sylabech se objevily názvy: imperativní, objektově orientované, procedurální a strukturované paradigma, jež jsou

[2] v rámci povinné výuky se seznámí se 4 programovacími jazyky a k nim si musí zvolit jeden ze tří povinně volitelných

vzájemně v jistém vztahu – imperativní paradigma můžeme považovat za zastřešující pojem, který je popsán v kategorii 2.3. Na ostatní vyjmenovaná paradigmata můžeme nahlížet jako specifické druhy imperativního paradigmatu. Z výsledků lze jednoznačně označit jako nejpreferovanější paradigma objektově orientovaného programování, které je součástí výuky na 10 fakultách. Toto zjištění není překvapující vzhledem k tomu, že ve výuce nejpoužívanější programovací jazyk *Java* ve výuce nejpoužívanější, je na objektově orientované programování zaměřen. Jelikož v některých zkoumaných předmětech zaměřených na výuku programování není uvedeno použité programovací paradigma, nemůžeme s jistotou porovnat množství vyučovaných paradigmat v rámci jednoho studijního programu. Pokud předměty s nedefinovaným paradigmatem neuvažujeme, v rámci jednoho studijního programu se vyučují 1–3 paradigmata.

Tabulka 4.2: Porovnání zastoupení programovacích paradigmat ve výuce

	PedF JU	PřF JU	FI MU	PřF OU	PedF OU	FP TUL ZŠ	FP TUL SŠ	PřF UHK ZŠ	PřF UHK SŠ	PřF UJEP	MFF UK	PedF UK	PřF UP	FAI UTB	PedF ZČÚ
neurč	0	**1**	**1**	0	**2**	**3**	**2**	**1**	**1**	0	1+**2***	**1**	0	0	0
funkcionální	0	0	**0,5**	0	0	0	**0,5**	0	0	0	**0,5**	0	**1,5**	0	0
imperativní	0	0	0	0	0	0	0	0	0	**0,5**	0	0	**0,5**	0	0
logické	0	0	**0,5**	0	0	0	**0,5**	0	0	0	**0,5**	0	0	0	0
OOP	**1**	**1**	**2***	**1**	0	**2**	**2**	**0,5**	**0,5**	**1,5**	1+**1***	**1**	0	0	**1**
procedurální	0	0	0	0	0	0	0	0	0	0	0	0	**2**	0	**1**
strukturované	0	0	0	0	0	0	0	**0,5**	**0,5**	0	0	0	0	0	0

4.5.3 Metodika výuky programování

V kapitole 2.6 byly popsány některé metodiky používané při výuce programování. Jedním z dílčích cílů výzkumu bylo zjistit, zda jsou některé z těchto metodik použity při výuce. Texty sylabů byly zkoumány na výskyt názvů těchto metodik. Podle výsledků pouze dva sylaby ze všech zkoumaných obsahovaly název použité metodiky a to předmět Programování 1 na PedF JU a předmět Objektové programování I na PřF JU. Oba tyto předměty používají stejné metodiky Object first a Design pattern first. První jmenovaná metodika je popsaná v kapitole 2.6, druhá jmenovaná klade při výuce důraz na výuku s důrazem na tzv. návrhové vzory. V žádném ze sylabů ostatních předmětů zaměřených na výuku programování není metodika zmíněna, nemůžeme tedy jednotlivé studijní programy porovnat.

4.5.4 Didaktika programování

Jelikož jsou zkoumáné programy zaměřené na výuku budoucích učitelů informatiky, bylo cílem zkoumat nejen odborné předměty zaměřené na výuku programování,

ale porovnat jednotlivé programy i v oblasti didaktiky programování. V rámci 1-fáze výzkumu výzkumu byla vytvořena kategorie DP, která zahrnuje všechny předměty, jejichž součástí byla výuka didaktiky programování, didaktiky algoritmizace, dětských programovacích jazyků a robotiky (počty těchto předmětů v rámci jednotlivých fakult najdeme v tabulce 3.3). V dalším kroku byly vybrány jen ty předměty, jejichž součástí je didaktika programování – v jejich názvu či obsahu se objevuje sousloví didaktika programování. Tyto vybrané předměty byly rozděleny do dvou skupin – na předměty jež se specializují výhradně na didaktiku programování (v tabulce popsány jako samostatně) a na předměty jež je výuka programování pouze dílčí součástí (nesamostatně).

Tabulka 4.3: Porovnání zastoupení didaktiky ve výuce

	PedF JU	PřF JU	FI MU	PřF OU	PedF OU	FP TUL ZŠ	FP TUL SŠ	PřF UHK ZŠ	PřF UHK SŠ	PřF UJEP	MFF UK	PedF UK	PřF UP	FAI UTB	PedF ZČÚ
samostatně	**1**	**2**	0	0	0	**1**	0	**1**	**1**	0	**1**	0	**1**	0	**1**
nesamostatně	**2**	0	**2**	**1**	0	**1**	**1**	0	0	0	0	**1**	0	**1**	**1**

Podle výsledků se tématu věnují všechny fakulty, v jejichž nabídce je i NMgr. obor učitelství. Právě 5 studijních programů obsahuje didaktiku programování pouze jako samostatný předmět, zatímco v 5 programech je didaktika programování pouze součástí didaktiky informatiky. Ve 3 studijních programech najdeme didaktiku programování jak samostatně, tak v rámci didaktiky informatiky. Patrné rozdíly jsou i v celkovém počtu předmětů v rámci jednoho studia. Na některých fakultách se setkají studenti s didaktikou programování pouze v rámci didaktiky informatiky, ale například na PřF JU jsou didaktice programování vyhrazeny dva samostatné předměty.

4.5.5 Robotika a unplugged teaching ve výuce

S programováním a algoritmizaci souvisí výuka dalších specifických témat, jako je robotika a unplugged teaching. V rámci 1. fáze výzkumu byly do kategorie DP zahrnuty i ty předměty, jejichž sylaby obsahují ve svém názvu či obsahu pojmy robotika nebo unplugged teaching, případně slova odvozená. Jelikož bylo zjištěno, že pojem unplugged teaching se nevyskytuje ani v jednom z analyzovaných předmětů, byly sylaby analyzovány pouze na výskyt pojmu "robotika". V rámci této analýzy byly sylaby předmětů opět rozděleny do dvou skupin na ty, jež se zaměřovaly výhradně na robotiku a na ty, u níž byla robotika pouze součástí.

Jak bylo zmíněno v kapitole 2.5, programování nemá u studentů učitelství velkou oblibu, což by mohly fakulty reflektovat mimo jiné i zařazením robotiky do výuky, na což poukazuje Nagyová (2014). Jak na to reagují fakulty? Podle výsledků není robotika v programech připravující učitele příliš rozšířená, vyučuje se pouze na 5 fakultách. Pouze na dvou fakultách (PedF JU, PedF UK) se robotika vyučuje jako

Tabulka 4.4: Porovnání zastoupení robotiky ve výuce

	PedF JU	PřF JU	FI MU	PřF OU	PedF OU	FP TUL ZŠ	FP TUL SŠ	PřF UHK ZŠ	PřF UHK SŠ	PřF UJEP	MFF UK	PedF UK	PřF UP	FAI UTB	PedF ZČÚ
samostatně	1	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0
nesamostatně	0	0	0	**1**	0	0	0	0	0	0	**1**	**1**	0	**2**	0

samostatný předmět, na PedF UK se objevuje i v rámci dalšího předmětu. Na třech fakultách najdeme robotiku pouze jako součást dalšího předmětu.

4.6 Koncepty výuky programování

V předchozí kapitole jsme prezentovali výsledky kvantitativního výzkumu, který porovnával fakulty v počtu předmětů zaměřených na oblast výuky programování. Při bližším zkoumání můžeme vysledovat určité koncepty v přístupu programování na jednotlivých fakultách. Představme si nejdříve metodologii hledání těchto konceptů a následně koncepty samotné.

Pro každou kombinaci studijních programů bakalářského a magisterského stupně studia byla sestavena posloupnost předmětů z kategorie výuky programování (VP) a didaktiky programování (DP)[3] podle jejich chronologického doporučeného zařazení do průběhu studia. Tyto posloupnosti předmětů byly vzájemně porovnávány vzhledem k různým parametrům – počtu vyučovaných programovacích jazyků, vyučovanému paradigmatu, podílu didaktických předmětů, možnosti zvolit si programovací jazyk. Z tohoto porovnání vzešly určité koncepty výuky programování, jež mohou být pojmenovány a popsány. Rozbory jednotlivých konceptů se mohou zdát strohé, ale pro zachování objektivity nelze bez dalšího zkoumání hodnotit dopad výuky jednotlivých konceptů na studentovu znalost v oblasti programování

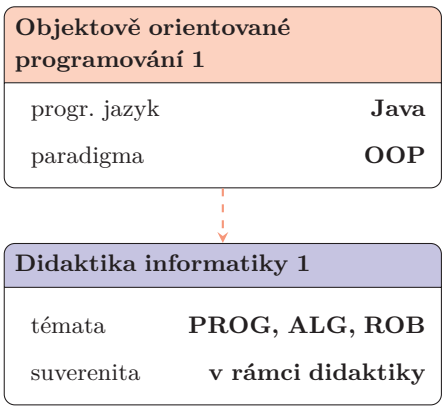
Ke každému konceptu je také pro přehlednost přidáno vizuální schéma jeho reálného použití. Přechody mezi bakalářským stupněm studia a navazujícím stupněm jsou označeny červenou čárkovanou čarou, barevně jsou odlišeny kategorie předmětů.

4.6.1 Minimalistický koncept

Minimalistický koncept představuje výuku programování v nejužší formě tak, aby studenti získali obecné povědomí o programování, aniž by byly tyto znalosti dál rozvíjeny. Příkladem takového konceptu je výuka na PřF OU, jež obsahuje programování pouze ve dvou předmětech. Prvním je předmět z kategorie programování s názvem Objektově orientované programování 1, který se dle sylabu zaměřen úzce na koncepty OOP, aniž by zde byly zmíněny koncepty strukturovaného progra-

[3]Pro přehlednost celé studie nebyly kategorie VA a WWW zařazeny, výzkum cílí na výuku programování.

mování. Druhým je předmět Didaktika informatiky 1, v jehož rámci je vyučována didaktika programování a robotika



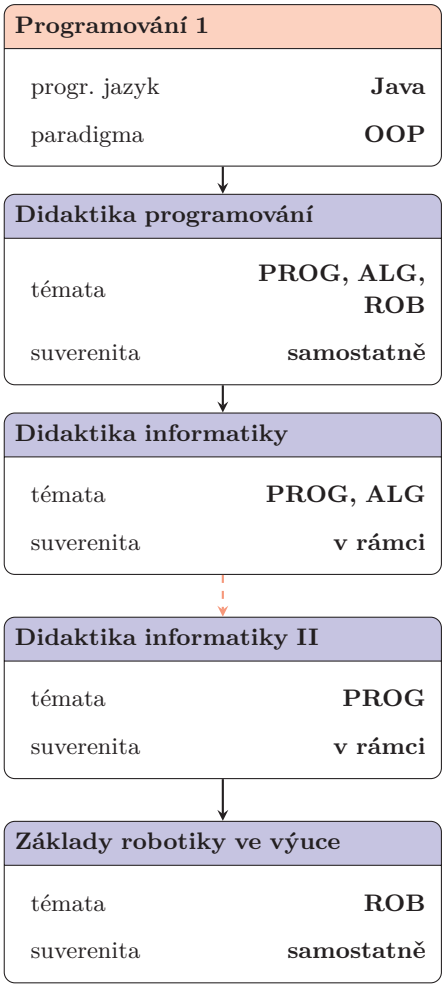
Obrázek 4.1: Schéma minimalistického konceptu

4.6.2 Didaktický koncept

Didaktický koncept klade důraz na výuku didaktiky programování, kterou upřednostňuje před programováním jako odborným předmětem. V tomto konceptu najdeme převahu předmětů zaměřujících se na didaktiku programování, dětské programovací jazyky a robotiku, při čemž dochází k upozadění odborné znalosti programování. Příkladem tohoto konceptu je výuka na PedF JU, v rámci které studenti absolvují ve 2. semestru bakalářského studia předmět Programování I, jež se zaměřuje na základy objektového programování, načež ostatní předměty jsou výhradně didaktické. Ještě v rámci bakalářského studia probíhá výuka dvou předmětů s názvem Didaktika programování a Didaktika informatiky. První z nich se zaměřuje jak je z názvu patrné výlučně na didaktiku programování, v druhém pak najdeme výuku programování a robotiku jako dílčí téma. V navazujícím studijním programu studenti absolvují předmět Didaktika informatiky II, jehož dílčím tématem je didaktika programování. Poslední předmět související s výukou programování nese název Základy robotiky ve výuce, je vyučován ve 3. semestru NMgr. studia a je vymezen výhradně na práci s robotickými stavebnicemi.

4.6.3 Přehledový koncept

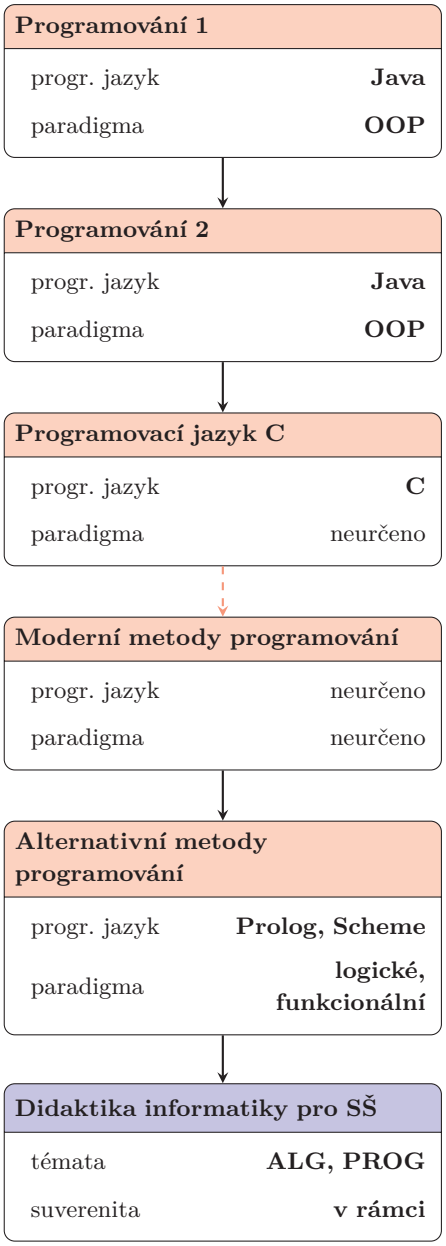
Další model výuky programování cíli na představení širokého množství programovacích jazyků a programovacích paradigmat. Převažují v něm předměty odborného programování nad didaktickými. Jednotlivým programovacím jazykům je věnován převážně jen jeden semestr. Příklad takového modelu najdeme na FP TUL, konkrétně v kombinaci Bc. programu s NMgr. programem pro výuku na SŠ. V této kombinaci najdeme celkem 5 odborných předmětů zaměřených na výuku programování a jeden didaktický. V bakalářském stupni studia studenti absolvují nejdříve dva předměty zaměřené na OOP a programovací jazyk *Java* s názvem Programování 1



Obrázek 4.2: Schéma didaktického konceptu

a Programování 2. Na konci bakalářského stupně je pak přidán další programovací jazyk v předmětu Programovací jazyk C. V navazujícím studiu studenti absolvují předmět s neurčitým názvem Moderní metody programování, jehož cílem je podle sylabu „ poskytnout studentům představu o technikách a přístupech při vytváření aplikací v prostředí Windows“. Sylabus tohoto předmětu neobsahuje použitý programovací jazyk či paradigma[4] a předmět Alternativní metody programování[5], ve kterém se se studenti setkají s programovacími jazyky *Prolog* a *Scheme*, jež sou zástupci logického a funkcionálního paradigmatu. Jediným zástupcem didaktických předmětů je Didaktika informatiky pro SŠ, kde se vyučuje didaktika programování jako dílčí téma.

[4]V akademickém roce 2015/2016 byl použit programovací jazyk C#.
[5]Sylabus neuvádí, podle čeho se hodnotí „alternativnost“ jazyka.

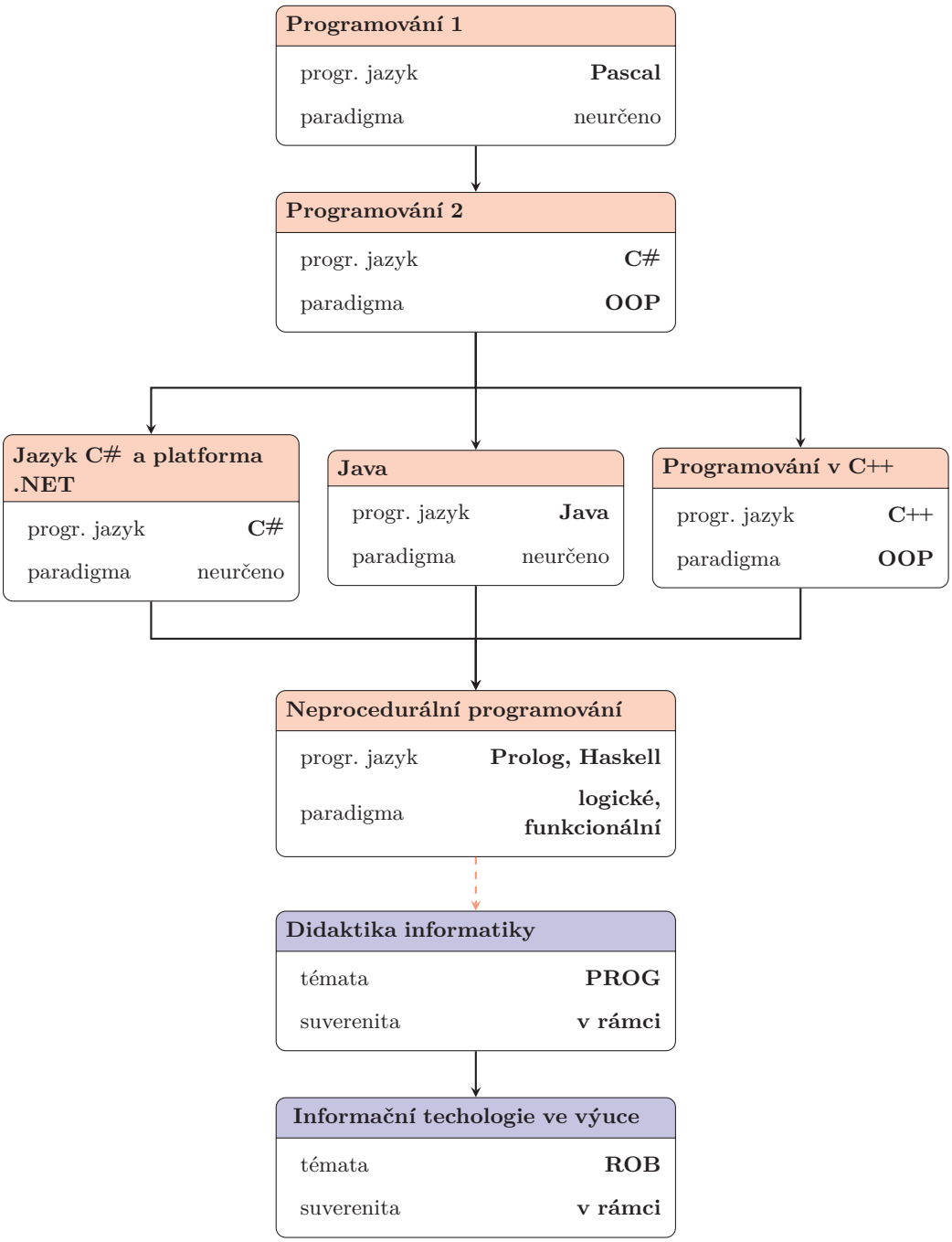


Obrázek 4.3: Schéma přehledového konceptu

4.6.4 Koncept volitelného předmětu

Některé programy dovolují studentovi v určité fázi studia vybrat programovací jazyk, jemuž se chce věnovat. Jako příklad uveďme studijní program MFF UK, kde si studenti mohou po absolvování dvou povinných předmětů Programovaní 1 (vyučován programovací jazyk *Pascal*) a Programování 2 (vyučován programovací jazyk *C#*) zvolit z nabídky 3 programovacích jazyků – *C#*, *Java* a *C++*. Po absolvování jednoho z těchto předmětů studenti na MFF UK pokračují studiem předmětu Neprocedurální programování, ve kterém se setkají s programovacími jazyky *Prolog* a *Haskell*, zástupci logického a funkcionálního programování. V NMgr. studiu se

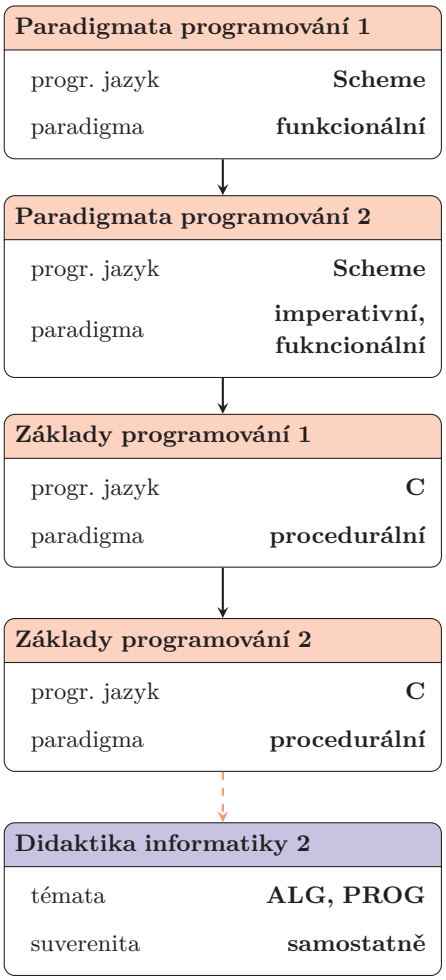
programování týkají pouze dva didaktické předměty, a to Didaktika informatiky, jejímž dílčím tématem je didaktika programování a předmět Informační technologie ve výuce, v jehož rámci najdeme výuku robotiky.



Obrázek 4.4: Schéma konceptu volitelného předmětu

4.6.5 Kvalitativní koncept

Poslední rozeznaný koncept necílí na množství představených programovacích jazyku, ale zaměřuje se na probraní menšího počtu programovacích jazyků do větší hloubky. Příkladem takového konceptu je výuka na PřF UP, která vyhrazuje pro oblast programování v bakalářském programu 4 odborné předměty, které jsou ale zaměřeny pouze na dva programovací jazyky. Nejdříve student absolvuje předměty Paradigmata programování 1 a Paradigmata programování 2, které jako výukový používají programovací jazyk *Scheme*. V prvním jmenovaném se probírá pouze funkcionální paradigma, v druhém se přidává i imperativní paradigma. Využití jazyka *Scheme* a funkcionálního paradigmatu v úvodním kurzu programování je v rámci ČR jedinečné, tento přístup nevyužívá žádná jiná fakulta. V dalších dvou předmětech Základy programování 1 a Základy programování 2 je pak k výuce použit programovací jazyk *C* a procedurální paradigma. V NMgr. pak studenti absolvují předmět Didaktika informatiky 2, který je ale celý vyhrazen didaktice programování a algoritmizace.



Obrázek 4.5: Schéma kvalitativního konceptu

5 Návrh konceptu pregraduální přípravy

V předchozích kapitolách jsme se seznámili s obecným přehledem výuky programování v ČR a představili jsme si některé rozeznatelné koncepty, které používají fakulty pří výuce budoucích učitelů informatiky. V této kapitole se pokusíme poznatky syntetizovat a navrhnout takový koncept pro výuku programování na fakultách vzdělavající budoucí učitele, jež by podle dosavadních zjištění byl nejvhodnější. Návrh se zaměřuje primárně na oblasti, které byly v rámci výzkumu detailně zkoumány – výuku programování a didaktiky programování ostatní kategorie – algoritmizace a programování v prostředí WWW – budou jen krátce komentovány.

Text této kapitoly bude mimo jiné záměrně psán v první osobě, protože se dostáváme do oblasti, kde nemůžeme zaručit stoprocentní objektivitu. Ačkoliv se budu odkazovat na některé výzkumy či osobní zkušenosti nemusí to znamenat, že bude pro každého čtenáře vybereme právě ten nejvhodnější koncept pregraduální přípravy. Stejně jako každý programátor může využít jiného programovacího jazyka k vytvoření programu se stejnou funkcí, může i fakulta „vytvořit“ kvalifikovaného studenta různými způsoby. Při šíři dnešní nabídky programovacích jazyků není možné naprosto objektivně rozhodnout, který z nich je pro výuku nejvhodnější, stejně tak jako nemůžeme jednoznačně rozhodnout, které programovací paradigma by mělo být vyučováno jako první. Každý čtenář musí zhodnotit, zda jsou pro něj prezentované argumenty relevantní, nebo ne.

Koncepty rozdělíme podle stupně studia na koncept pro bakalářské studium a následně i pro navazující magisterské studiu. V každé části budou představena jednotlivá specifika výuky, načež i navrhovaný koncept výuky.

5.1 Cíle výuky programování

Důležitou součást každého formálního vzdělávání je i stanovení výukových cílů, jinak řečeno představ o stavu, kterého má být dosaženo v určitém čase (Nezvalová 2006). Ačkoliv se tento termín používá převážně v souvistlosti se ZŠ a SŠ, použijeme ho nyní v kontextu vysokého školství a definujme si cíle, kterých by měl dosáhnout náš koncept výuky programování v 5-letém horizontu.

Cílem konceptu je připravit studenta na výuku algoritmizace a programování na základních a středních školách. Důraz je kladen na porozumění algoritmických principů využívaných při programování a jejich schopnost přenášet je do výuky na ZŠ a SŠ. Student po dokončení studia:

- samostatně navrhuje a programuje pokročilé programy s grafickým uživatel-

ským rozhraním programy alespoň v jednom programovacím jazyku;

- rozeznává různá paradigmata programování, je schopen vysvětlit jejich odlišnost, vytváří programy v rámci procedurálního, objektově orientovaného a funkcionálního paradigmatu;
- má přehled o trendech v oblasti výuky programování a dětských programovacích jazyků, je schopen vytvořit výukovou hodinu za pomoci alespoň jednoho dětského programovacího jazyku.
- hodnotí nové trendy ve výuce programování a algoritmizace, kontstruuje vyučovací hodinu s použitím těchto metod

Proč je vhodné definovat si cíle výuky v pregraduální přípravě, ačkoliv to není běžné? Výsledky našeho výzkumu ukazují na velké rozdíly ve výuce programování mezi jednotlivými fakultami. Proč jsou mezi fakultami takové rozdíly, když všichni absolventi těchto fakult budou učit podle stejného rámcového programu pro ZŠ nebo SŠ? Dle mého názoru v ČR chybí platforma, která by definovala výstupní znalosti absolventa učitelství, nebo by zde probíhala alespoň diskuze na celorepublikové úrovni. Kladu si otázku, zda značná volnost pro fakulty při skladbě studijního programu, zbytečně nediskriminuje absolventy některých fakult? Uveďmě si jako příklad PřF OU, kde studenti v součtu bakalářského i navazujícího magisterského studia absolvují výuku programování pouze v jednom předmětu. Jsou absolventi této fakulty dobře připraveni na výuku programování na základních a středních školách, když se jinde učí programování až v 5 předmětech? Tuto otázku samozřejmě bez patřičného výzkumu nelze zodpovědět, ale rád bych na tuto nerovnost upozornil.

Definice cílů přípravy budoucích učitelů informatiky, jak v oblasti programování, tak i v celkovém pohledu, by mohla pomoci i s formováním vědního oboru didaktiky informatiky, která podle Vaníčka a Černochové(2015) není v ČR ještě plně zavedená. Její stav popisují takto:

> „Profesní a odborná podpora učitelů informatiky a systém dalšího vzdělávání učitelů informatiky je podobně roztříštěná jako její didaktikové: na celostátní úrovni není vypracován a neprobíhá systém školení nebo konference učitelů, zaměřené na vyučování předmětu ICT, nemá žádného garanta (profesní sdružení Jednota školských informatiků nemá masívní kontakt s učiteli z praxe). Možné příčiny izolovanosti učitelů informatiky spatřujeme i v nízké aprobovanosti učitelů, kteří se jako informatici často necítí a nemají tendenci se profesně vzdělávat, sdružovat a komunikovat.“(Černochová – Vaníček 2015, s. 184)

Jak by se ale mohli učtelé informatiky sdružovat, když jejich identita jejich oboru není společností definována[1]?

[1]Definovat učitele informatiky pouze jako absolventa učitelského programu je poněkud nedostačující.

5.2 Výuka na bakalářském stupni studia

Podle výsledků 1. fáze výzkumu jsou v bakalářském studiu vyučovány převážně odborné předměty programování a algoritmizace, zatímco v navazujícím studiu se vyučují hlavně předměty didaktické. Toto rozdělení hodnotím jako smysluplné – zatímco pro studium úvodu do programování nepotřebuje student odborné znalosti na úrovni na VŠ, v didaktice programování je nepochybně výhodné navazovat na předchozí znalosti z výuky programování a také obecné didaktiky. Studenti mohou také při těchto didaktických předmětech reflektovat zkušenosti ze svých praxí na ZŠ a SŠ. Dále je i takováto konzistence v rozmístění předmětů vhodnější při přestupu na NMgr. studium na jinou fakultu nebo univerzitu, proto v připravovaném modelu navrhuji vyhradit bakalářské studium pro výuku odborných předmětů výuky programování a navazující magisterské pro výuku didaktických předmětů. Dále se tak v textu o bakalářském stupni studia omezíme na odborné předměty zaměřené na výuku programování.

5.2.1 Úvodní programovací jazyk

Důležitou roli ve výuce programování hraje i výběr programovacíh jazyků, které budou studentovi představeny. Výuka programování by se neměla zaměřovat jen na výuku znalosti programování v určitém programovacím jazyku, ale na pochopení programovaní jako celku, a to nezávisle na zvoleném programovacím jazyku. Proto je vhodné volit jazyky s jednoduchou syntaxí a celkovou přehledností, tak aby výuka specifik tohoto jazyka nepřevažovala nad výukou obecných principů programování.

V kapitole 2.6 jsme definovali vlastnosti, které by měl mít programovací jazyk vhodný pro výuku programování. Jelikož ale jsou naší cílovou skupinou budoucí učitelé, je vhodné tyto požadavky rozšířit o aspekt využitelnosti ve školní praxi – jazyk by měl být využívaný (nebo alespoň použitelný) pro výuku na ZŠ nebo SŠ, aby student mohl jeho znalost zužitkovat ve své praxi.

Problematikou výuky programování na VŠ se zabýval Mareš, který jako idealní považuje při vyučování použít dva programovací jazyky,

> „jeden pro základní kurs programování (pravděpodobně stačí jednosemestrální), ve kterém se studenti naučí úplné základy. Na těch pak mohou stavět jak teoretičtější přednášky o algoritmech a datových strukturách, tak výuka dalšího, pokročilejšího jazyka. Jako základní jazyk se podle mého názoru nejvíce hodí buďto starý dobrý Pascal nebo interaktivní jazyk typu Python. Pro pokročilejší kursy se pak nabízí buďto zůstat u Pascalu a rozšířit ho na Object Pascal, nebo se přesunout k *C* a *C#*. Jelikož se ale požadavky jednotlivých oborů liší, mohlo by být šikovné od tohoto bodu výuku dalších jazyků „modularizovat, tj. učinit z nich volitelné předměty, přičemž pro každý obor bude nějaká podmnožina z nich povinná.“ (Mareš 2007, s. 184, upraveno)

Programovací jazyk Python, který Mareš zmiňuje, v současnosti získává na popularitě jak ve výuce na základních a středních školách (využívá se například na

Slovensku (Hájek – Peter 2015), Chorvatsku (Mladenović et al. 2016), Maďarsku (Mészárosová – Tomcsányiová 2015) nebo Finsku (Grandell et al. 2006), tak i na VŠ (jedná se v současnosti nejvíce využívaný programovací jazyk na předních amerických univerzitách (Guo 2014), včetně slavné MIT, používá se při výuce na Slovensku (Blaho – Tomcsányi 2015, Mészárosová – Tomcsányiová 2015) , je nejpoužívanějším výukovým jazykem v Austrálii a Novém Zélandě (Mason – Cooper 2014) nebo je druhým nejpouživanějším v Anglii (ale jeho podíl roste) (Murphy et al. 2016)

Výhody Pythony pro výuku programování zmiňuje Meszaroszová (2015):

- Jedná se o multiplatformní jazyk, programy napsané v Pythonu můžeme spustit na všech dnes běžně používaných operačních systémech.
- Vyniká vysokou čitelností kódu, je přehlednější než ostatní jazyky.
- Je k dispozici velké množství knihoven, které mohou rozšířit možnosti jazyka, napří knihovna Tkinter, která umožňuje vytvářet programy s grafickým rozhraním nebo knihovna Turtle, která umožňuje vytvářet„želví grafiku“, podobně jako v programovacím jazyku Logo.
- Nejedná se jen o výukový jazyk, je možné ho použít pro vědecké účely, tvorbu webových stránek či počítačových her.

Některá negativa Pythonu popsal Zelle (Zelle 1999):

- Jelikož se jedná o iterpretovaný jazyk, chybí zde proces kontroly datových typů během kompilace (může dojít k chybě v programu, pokud je některé z hodnot přiřazen neočekávaný datový typ).
- Není tak rychlý, jako kompilované jazyky.

Python není dokonalý jazyk (žádný programovací jazyk není dokonalý), ale z výzkumů vyplývá, že se jedná v současnosti o jeden z nejvhodnějších jazyků pro úvodní kurz programování(2015) zatímco spňuje i kritéria vhodného jazyku pro výuku, která byla definována v kapitole 2.6 – je multiplaformní, zdarma a používá se i v komerční sféře. ([2] Je zajímavé, že ačkoliv se v tak velkých státech jako je USA, Anglie nebo Australie se Python spolu s Javou řadí k nejčastěji vyučovaným programovacím jazykům v úvodních kurzech programování, v rámci přípravy učitelů informatiky v ČR se vyskytuje tento předmět pouze na jedné fakultě, v jednom z předmětů.

Ačkoliv Mareš navrhuje, aby byl úvodní programovací jazyk použit pouze v jednom semestru, navrhuji použít pro učitelské obory minimálně dvousemestrální výuku. Jelikož by Python mohli absolventi využít i při své budoucí praxi, je vhodné tomuto jazyku věnovat čas a umožnit studentům dostatečně jazyk „zažít“. V prvním semestru by se naučili základům programování a v druhém už vytvářet programy

[2] Osobitý názor nabízí na svém blogu Lessner: „Nejlepší programovací jazyk pro začátečníky je jednoznačně a bez debaty Python, především proto, že ho mám nejraději, a je nás víc.“

s grafickým uživatelským rozhraním[3]. Dále by se mohl využít tento jazyk v některém předmětu na výuku algoritmů a datových struktur jako to dělají v předmětu Algoritmy a dátové štruktúry na FMFI UK v Bratislavě (Blaho – Tomcsányi 2015).

5.2.2 Další programovací jazyk

Ačkoliv je jazyk Python multiparadigmatický a je možné s jeho použitím vyučovat procedurální, objektově orientované i funkcionální paradigmatu, výuka pouze v programovacím jazyku Python by byla značně monotématická (všechna paradigmata by se pravděpodobně vměstnat do dvousemestrálního kurzu programování) proto je vhodné do výuky zařadit i další programovací jazyky, ve kterých studenti získají zkušenosti s dalšími paradigmaty a mohou s nimi Python kriticky porovnat. Jelikož by studenti měli zažité základy programování z Pythonu, mohli by se soustředit poze na specifika pro ně nového jazyka a brzy samostatně tvořit složitější programy. Nabízí se několik více či méně obvyklých programovacích jazyků a jejich specifik:

- C – kompilovaný, nízkoúrovňový jazyk,
- C++ – výuka OOP, kompilovaný jazyk,
- Java – výuka OOP, možnost využít výukové prostředí BlueJ nebo Greenfoot,
- Scheme, Haskell – výuka funkcionálního paradigmatu,
- Prolog – výuka logického paradigmatu,
- Javascript – výuka OOP, možnost pokračovat v některém předmětu zaměřeném na tvorbu webových stránek.

Pokud mají být naplněny definované cíle výuky, měli bychom zařadit předměty se všemi paradigmaty. Pro OOP se jeví jako nejvhodnější Java, pro kterou jsou k dispozici výuková vývojová prostředí specializovaná na pochopení principu OOP. Funkcionální paradigma by mohlo být zařazeno spolu s logickým programováním v jednom předmětu po vzoru některých fakult z našeho výzkumu. Pro funkcionální programování jako vhodný považuji jazyk Scheme, jež má jednoduchou syntaxi (Skoupil 2007a). Pro logické programování je možné použít jazyk Prolog. Další programovací jazyky by mohly být nabídnuty ve formě povinně volitelných předmětů.

5.3 Výuka na navazujícím stupni studia

5.3.1 Didaktika programování

Didaktické předměty jako je didaktika informatiky, didaktika programování či edukační robotika odlišují přípravu budoucích učitelů informatiky od přípravy budoucích programátorů. Didaktika informatiky by měla studenty připravit na to, co, jak

[3]Pokud by se v praxi ukázalo, že se studenti nestíhají za dva semestry výuky samostatně programovat aplikace s grafickým uživatelským rozhraním, byla by výuka rozšířena na 3 semestry.

a proč v informatice vyučovat. Existence samostatných předmětů didaktiky programování dává tušit, že má tato oblast pro učitele informatiky velký význam a zároveň je její obsah natolik rozsáhlý, aby se oddělila od didaktiky informatiky (Salanci 2012). Salanci zdůvodnil, proč má didaktika programování své místo v přípravě učitelů informatiky:

> „Poznatky z didaktiky programovania dávajú učiteľovi informatiky do rúk veľmi silný nástroj na to, aby správne a zaujímavo realizoval tematický celok algoritmy a riešenie problémov, či korektným spôsobom v žiakoch rozvíjal algoritmické myslenie. Poznatky z didaktiky programovania poskytujú učiteľovi iný pohľad na to, prečo je potrebné dodržiavať pri vyučovaní určité zásady. Dávajú tiež odpovede na to, prečo majú žiaci ťažkosti s porozumením nových poznatkov.“ (Salanci 2012, s. 212)

Jak jsme popsali v teoretické části, poslední trendy ve výuce informatiky ukazují, že algoritmické myšlení a programování získává ve výuce stále větší podíl, vysokoškolská příprava učitelů informatiky by na to měla co nejdříve reagovat, proto navrhuji zařadit samostatnou výuku didaktiky v jednom semestru. Tento předmět by nenahradil didaktiku informatiky, pouze by dovolil se jednotlivým tématům věnovat více do hloubky.

5.3.2 Výukové programovací jazyky

Další oblastí, ve které by se měla výuka budoucích učitelů lišit od výuky programátorů je výuka tzv. výukových programovacích jazyků. Jako výukový můžeme nazvat jazyk, který se hodí hlavně pro výuku programování, ale nemusí být vhodný pro tvorbu komerčních programů. Do této skupiny zařazujeme např. programovací jazyky Scratch, Logo, Baltík nebo Karel. V současnosti jsou populární tzv. vizuální programovací jazyky (např. Scratch nebo Baltík), jejichž zdrojový kód je reprezentován formou obrázků. Díky atraktivnímu grafickému prostředí jsou jazyky jsou využívané pro úvodní kurzy programování základních školách. Ačkoliv se tyto jazyky mohou zdát„dětské“, například Scratch se používá při výuce programování několik amerických univerzitách (Guo 2014). Tyto jazyky jsou dle mého názoru nedílnou součástí výuky programování a studenti by s nimi měli být obeznámeni. Nejedná se o žádnou novinku, například programovací jazyk Logo vznikl už v roce 1967.

Jelikož už jsou studenti na navazujícím studiu obeznámeni s teorií programování a zároveň výukové jazyky nemají složitou syntaxi na naučení, není nutné tomuto tématu věnovat samostatný předmět. Jako smysluplnější se jeví vyučovat toto téma v rámci didaktiky programování a to některou z forem, při níž se do výuky studenti aktivně zapojí – například by si mohl každý nastudovat jeden jazyk samostatně a následně ho při hodině předvést svým spolužákům a společně by diskutovali nad kvalitami či možnostmi použití daného jazyka.

5.3.3 Robotika

V poslední době se můžeme setkat ve vyučování informatiky na základních a středních školách s využíváním robotických stavebnic, jakožto zajímavým způobem roz-

šíření výuky programování. Na rozdíl od dětských programovacích jazyků robotika rozvijí i manuální zručnost (Brodenec – Trhan 2012). Další specifika robotiky i s praktickými zkušenostmi popisuje Kalaš et al. (2014):

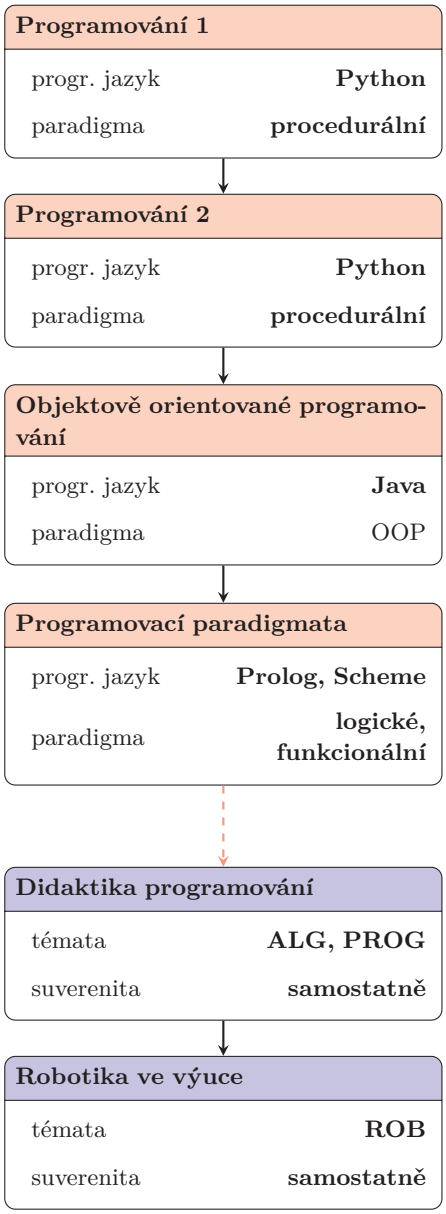
> „Programovanie v edukačnej robotike sa však odlišuje od klasického programovania – žiakom ponúka konkrétnu, bezprostrednú a hmatateľnú reprezentáciu či „vizualizáciu“ pre nich zvyčajne abstraktného programu. Počas našej praxe sme sa veľakrát stretli s prípadmi, kedy žiaci počas konštruovania modelu začali správne chápať programovanie ako prostriedok na dosiahnutie svojho zámeru (napr. rozhýbanie robotického modelu), nie ako samostatný cieľ.“

Robotika má pozitivní dopad na výuku a zvyšuje motivaci žáků k programování (STOFFOVA – TAKAC 2013), proto navrhuji vyhradit pro robotiku jeden samostatný předmět. Jelikož se v edukační robotice používají grafické programovací jazyky, je vhodné aby výuka následovala až na konci navazujícího studia.

5.4 Navrhovaný studijní plán

Představme si nyní navrhovanou koncepci výuky programování. Návrh je zpracován stejnou grafickou formou jako v případě analýzy koncepcí programování, ale je doplněn o další komentáře. V konceptu jsou zapracovány všechny výše zmíněné zjištění a návrhy. Součástí plánu jsou dva předměty zaměřené na výuku programování v jazyku Python, jejichž cílem v poslední fázi je naučit studenta samostatně programovat aplikace s GUI. Na tyto dva semestry navazují dva další předměty soužící k rozšíření studentových znalostí a osvojení dalších druhů paradigmat. V programu navazujícího studia doporčuji zařadit dva didaktické předměty zaměřené na didaktiku programování a robotiku ve výuce, tak aby absolvent měl dostatečné porfolio metod, aby mohl programování vyučovat.

Pokud porovnáme identifikované koncepty výuky s navrženým ideálním, nalezneme určité podobnosti ale ne úplnou shodu. Idealní program přebírá od kvalitatího snahu zdokonalit se v jednom programovacím jazyku, od přehledového množství programovacích jazyků a stejně jako v didaktickém konceptu se vyučují didaktika programování nebo robotika v samostatných předmětech.



Praktické i teoretické seznamování s programováním. Témata: řetězce, výstup na obrazovku, textové řetězce, cykly, pole, práce se souborem.

Práce s GUI, práce s dalšími knihovnami samostatný vývoj aplikací.

Výuka základů OOP v jazyce Java, využití některého z výukových prostředí(BlueJ, Greenfoot, porovnání procedurálního a objektově orientovaného paradigmatu

Výuka základů logického a funkcionálního paradigmatu, porovnání všech studentům známých paradigmat.

Problematika didaktické transformace učiva programování a algoritmizace na ZŠ a SŠ, možnosti uplatnění výukových programovacích jazyků při výuce, jejich provnání.

Představení v současnosti používaných robotických stavebnic, specifika výuky robotiky, tvorba samostatných projektů

Obrázek 5.1: Schéma navrhovaného konceptu

6 Závěr

Cílém této diplomové práce bylo analyzovat výuku programování v Česku akreditovaných studijních programech zaměřených na přípravu budoucích učitelů informatiky a jednotlivé programy porovnat. V teoretické části práce jsem na základě odborné literatury definoval pojmy algoritmizace a programování. Pro zjištění podoby výuky těchto témat na ZŠ a SŠ jsem analyzoval státem deklarované kurikulum a porovnal ho se zahraničním – slovenským kurikulem. Zdůvodnil jsem, proč se programování zařazuje do pregraduální přípravy i popsal způsoby, jakým lze výuku koncipovat. V rámci praktické části práce jsem získal studijní plány všech programů vzdělávající budoucí učitele informatiky, ve kterých jsem následně označil předměty, které souvisejí s programováním a algoritmizací. Tyto předměty jsem kategorizoval podle tématu do 4 skupin (výuka programování, teorie algoritmů, didaktika programování a programování v oblasti WWW), díky čemuž jsem mohl provést první porovnání jednotlivých programů, založené na četnosti předmětů v jednotlivých skupinách. Toto porovnání ukázalo na poměrně velké rozdíly ve způsobu výuky jednotlivých programů. Abych získal i důležitá data o obsahu jednotlivých předmětů, aplikoval jsem metodu textové analýzy na dvě ze zmíněných kategorií, díky čemuž jsem získal důležitá data jako např. použitý programovací jazyk nebo paradigma v jednotlivých předmětech zaměřených na výuku programování. Tyto údaje jsem využil pro porovnání jednotlivých programů. Na základě rozdílů v poměru didaktických a odborných předmětu nebo počtu vyučovaných programovacích jazyků jsem vytvořil a popsal koncepty výuky programování a uvedl příklady těchto konceptů. V poslední části práce jsem na základě dosavadního výzkumu definoval cíle pregraduální přípravy v oblast programování, abych následně mohl popsat doporučené způsoby, jak tyto cíle naplnit a vytvořit modelový výukový plán pro výuku programování, didaktiky programování, výukových programovacích jazyků a robotiky.

Podle výsledků jsou mezi jednotlivými fakultami výrazné rozdíly v součtu předmětů věnovaných všem zkoumaným tématům – 10 předmětů na MFF UK oproti 4 na PřF OU, což může mít dopad na kvalitu vzdělání absolventa v této oblasti. Velké rozdíly najdeme i v počtu předmětů věnovaných dílčí oblasti výuky programování – 5 v programech FP TUL pro ZŠ i SŠ oproti 1 na PedF JU nebo oblasti didaktických předmětů (didaktika programování, výuka dětských programovacích jazyků nebo edukační robotiky) – 4 na PedF JU oproti 1 na PřF UP. Fakulty se naopak shodují v rozložení předmětů do průběhu studia – v bakalářském studiu probíhá výuka programování a algoritmizace, v navazujícím studiu pak výuka didaktických předmětů. Jako nejčastěji používaným jazyky pro výuku programování byla identifikovaná Java, Výuka programování není v ČR jednotná, v součtu všech

vyučuje se minimálně 11 různých programovacích jazyků v rámci povinných předmětů. Jako nejčastěji využívaným programovacím jazykem byla vyhodnocena Java, fakulty pro výuku používají převážně objektově orientované paradigma. Ve výuce programování a didaktických předmětů bylo rozeznáno 5 různých konceptů výuky – minimalistický, didaktický, přehledový, modulární a kvalitativní. Za stěžejní v celé práci považuji zjištění nejednotnosti v přístupu k výuce programování a algoritmizace napříč fakultami. S tímto faktem jsem pracoval v poslední části své práce a definoval výukové cíle, kterých by měla dosáhnout pregraduální příprava učitelů informatiky. Při následném hledání ideální cesty k naplnění těchto cílů jsem vyhodnotil programovací jazyk Python jako nejvhodnější pro výuku programování. Doporučuji výuce v tomto jazyku vyhradit minimálně dva semestry, aby mohlo dojít k dostatečnému osvojení. Zdůrazňuji také zařazení výuky didaktiky programování, která je důležitá pro osvojení principů didaktické transformace oblasti programování a zařazení výuky edukační robotiky, jež může výuku programování zatraktivnit.

Nejsložitější pro mě byl návrh konceptu kategorizace předmětů do 4 skupin a jeho následné provedení. Obtížné bylo rozdělení předmětů do kategorií „výuka programování“ a „výuka algoritmizace“, protože předměty v sobě integrovaly prvky z obou těchto kategorií, přesto vnímám tyto dvě kategorie jako odlišné, kladou si různé cíle. Situaci komplikovala i různá struktura popisu v sylabech předmětů napříč fakultami. Do dnešního dne jsem neobjevil lepší způsob kategorizace předmětů

Jako hodnotné na své práci považuji vytvoření a popis konceptů výuky programování na českých fakultách. V dalším výzkumu by bylo možné zkoumat, zda má výběr tohoto konceptu vliv například na výuku absolventů – zda mají k výuce programování kladný vztah nebo jaká témata do výuky zařazují. Důležitým počinem bylo také definování cílů výuky pregraduální přípravy a vytvoření konceptu výuky. Tyto cíle by mohly být podstoupeny odborné diskuzi mezi didaktiky informatiky či vedoucími příslušných kateder, rád se se dozvím jejich názor.

7 Použité zdroje

Beck, Kent et al, 2001. *Manifest Agilního vývoje software* [online]. Dostupné z: `http://agilemanifesto.org/iso/cs/principles.html`.

Berki, Jan, 2011. ICT in the Czech and Slovak National Curriculum. In *Proceedings of 5th Internaitonal Conference ISSEP*. Association of the Infovek Project and Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava., ISBN 978-80-89189-90-7.

Berki, Jan, 2013. Study programs of training teachers of informatics in the Czech republic. In *Journal of Technology and Information Education*, Dostupné z: `http://www.jtie.upol.cz/clanky_1_2013/JTIE-1-2013.pdf`.

Berki, Jan, 2016. *Projektované, realizované a dosažené ICT kurikulum na základních školách*. Disertační práce, Jihočeská univerzita v Českých Budějovicích, Pedagogická fakulta, České Budějovice, Dostupné z: `http://theses.cz/id/y5olfe/`.

Blaho, Andrej, Tomcsányi, Peter, 2015. Programovací jazyk Python vo výučbe predmetu Algoritmy a dátové štruktúry. *In Didinfo 2015*.

BlueJ, n.d. *Objects First With Java - A Practical Introduction Using BlueJ* [online].[cit. 14.03.2017]. Dostupné z: `https://www.bluej.org/objects-first/description.html`.

Bolshakova, Elena, 2005. Programming paradigms in computer science education. In *International journal information theories*. ISSN 1310-0513. Dostupné z: `http://www.foibg.com/ijita/vol12/ijita12-3-p13.pdf`.

Brodenec, Ivan, Trhan Peter, 2012. Aktivity Katedra informatiky FPV UMB pre podporu využovania robotických stavebnic. In *Didinfo 2012*, Dostupné z: `http://didinfo.umb.sk/public/filestore/documents/richtext/138/didinfo2012.pdf`.

Cormen, Thomas H., 2013. *Algorithms Unlocked*. MIT University Press Group Ltd, ISBN 978-0-262-51880-2.

Cormen, Thomas H. et al., 2009. *Introduction to Algorithms*. The MIT Press, ISBN 978-0-262-03384-8.

Černochová, Miroslava, Vaníček, Jiří, 2015. *Didaktika informatiky na startu*. In Oborové didaktiky : vývoj – stav – perspektivy. Brno: Masarykova univerzita, ISBN 978-80-210-7769-0.

Department for Education, 2013. *The National Curriculum in England: Key Stages 1 and 2 framework document*. Dostupné z: https://www.gov.uk/government/publications/national-curriculum-in-england-primary-curriculum

Dvořáková, Ilona, 2010. *Obsahová analýza / formální obsahová analýza / kvantitativní obsahová analýza*. ANTROPOWEBZIN. Dostupné z: http://www.antropoweb.cz/media/webzin/webzin_2_2010/Dvorakova__I-2-2010.pdf.

Fojtík, Rostislav, 2013. *Moderní přístupy k výuce programování*. JTIE. s. 58–62. ISSN 1803537X. doi: 10.5507/jtie.2013.008. Dostupné z: http://jtie.upol.cz/cz/artkey/jti-201301-0008.php.

Fojtík, Rostislav, 2015. Vzdělávání budoucích učitelů informatiky. In *Didinfo 2015*.

Gavora, Peter, 2015. Obsahová analỳza v pedagogickom vỳskume: Pohlad na jej súcasné podoby. In *Pedagogická orientace*. 25, 3, s. 345.

Gosling, James et al., 2014. *The Java Language Specification*, Java SE 8 Edition. Addison-Wesley Professional, 1. edice, ISBN 013390069X, 9780133900699.

Grandell, Linda et al., 2006. Why Complicate Things?: Introducing Programming in High School Using Python. In *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52*, ACE '06, s. 71–80, Darlinghurst, Austrálie, Australian Computer Society, Inc. Dostupné z: http://dl.acm.org/citation.cfm?id=1151869.1151880. ISBN 1-920682-34-1.

Greenfoot. 2016. *About Greenfoot* [online].[cit. 14.03.2017]. Dostupné z: https://www.greenfoot.org/overview.

Guo, Philip, 2014. *Python is Now the Most Popular Introductory Teaching Language at Top U.S. Universities* [online].[cit. 10.06.2017]. Dostupné z: https://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/fulltext

Harel, David, Feldman, Yishai, 2004. *Algorithmics: The Spirit of Computing (3rd Edition)*. Addison-Wesley, ISBN 0-321-11784-0.

Hájek, Zdeněk, Vasaráb, Peter, 2015. Skúsenosti s programovaním na základnej škole. In *Didinfo 2015*.

Kalaš, Ivan, 2017. ScratchMaths: vzdelávací obsah a princípy tvorby. In *Spojená medzinárodná konferencia DidInfo a DidactIG 2017*, s. 16–24. UMB Banská Bystrica,

Kalaš Ivan et al., 2014. *SPECIFIKACIA VZDELAVACICH CIELOV PRE EDUKACNU ROBOTIKU*. JTIE. 2014, 6, 1, s. 30–44. ISSN 1803537X. doi: 10.5507/jtie.2014.003. Dostupné z: http://jtie.upol.cz/cz/artkey/jti-201401-0003.php.

Knuth, Donald E., 2016. *Umění programování. 1. díl: Základní algoritmy*. Computer Press. ISBN 978-80-251-2025-5.

Kofune, Yasuyo, Koita, Takahiro, 2012. Method for Improving Students' Programming Skills. In *Proceedings of IMCIC - ICSIT 2014*. ISBN 978-1-941763-00-1.

Lessner, Daniel, 2015. *Povinné programování od první třídy — zamkněte děti doma!* [online].[cit. 14.05.2017]. Dostupné z: http://ucime-informatiku.blogspot.co.id/2015/10/povinne-programovani-od-prvni-tridy.html

Lessner, Daniel, 2013. *Výuka efektivity algoritmů na gymnáziích*. JTIE. s. 12–20. ISSN 1803537X. doi: 10.5507/jtie.2013.002. Dostupné z: http://jtie.upol.cz/cz/artkey/jti-201301-0002.phpl.

Leutenegger, Scott, Edgington, Jeffrey, 2007. A Games First Approach to Teaching Introductory Programming. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '07, s. 115–118, New York, NY, USA, ACM. Dostupné z: http://doi.acm.org/10.1145/1227310.1227352l. ISBN 1-59593-361-1.

Mannila, Linda et al., 2006. An Objective Comparison of Languages for Teaching Introductory Programming. In *Proceedings of the 6th Baltic Sea Conference on Computing Education Research: Koli Calling 2006*, Baltic Sea '06, s. 32–37, New York, NY, USA, ACM.Dostupné z: http://doi.acm.org/10.1145/1315803.1315811l.

Mareš, Martin, 2007. *Tři otázky k výuce programování* [online].[cit. 14.06.2017]. Dostupné z: http://mj.ucw.cz/papers/proglang.htmll.

Mason, Raina, Cooper, Graham, 2014. Introductory Programming Courses in Australia and New Zealand in 2013 - Trends and Reasons. In *Proceedings of the Sixteenth Australasian Computing Education Conference - Volume 148*, ACE '14, s. 139–147, Darlinghurst, Australia, Australia, Australian Computer Society, Inc. Dostupné z: http://dl.acm.org/citation.cfm?id=2667490.2667507l. ISBN 978-1-921770-31-9.

Mészárosová, Eva, 2015. Is Python an Appropriate Programming Language for Teaching Programming in Secondary Schools? In *International Journal of Information and Communication Technologies in Education.*, s. 5–14. Dostupné z: https://periodicals.osu.eu/ictejournal/dokumenty/2015-02/ictejournal-2015-2-article-1.pdfl.

Mladenović, Monika et al. 2016. Introducing Programming to elementary Students Novices by using Game Development in Python and Scratch. In *EDULEARN16 Proceedings, 8th International Conference on Education and New Learning Technologies*, s. 1622–1629. IATED, 4-6 July. Dostupné z: http://dx.doi.org/10.21125/edulearn.2016.1323l. ISBN 978-84-608-8860-4.

MŠMT, 2008. *Rámcový vzdělávací program pro odbor vzdělávání: 18 – 20 – M/01 Informační technologie* [online]. [cit. 19. 4. 2017]. Dostupné z: http://www.nuv.cz/file/159_1_1/l.

MŠMT, 2016. *Rámcový vzdělávací program pro základní vzdělávání* [online]. [cit. 19. 4. 2017]. Dostupné z: http://www.nuv.cz/uploads/RVP_ZV_2016.pdfl.

Motyčka, Arnošt, 1999. *Algoritmizace*. Brno: Konvoj. ISBN 80-85615-80-0.

Mészárosová, Eva, Tomcsányiová, Monika. 2015. Vyučovanie základov programovania v jazyku Python na niektorých SŠ a VŠ. In *Didinfo 2015*,

Murphy, Ellen et al., 2016. *An Analysis of Introductory University Programming Courses in the UK*. CoRR. , Dostupné z: http://arxiv.org/abs/1609.06622l.

Musilek, Michal, 2013. *Project Scratch*. JTIE. 5, 1, s. 102–106. ISSN 1803537X. Dostupné z: http://jtie.upol.cz/artkey/jti-201301-0015.php.

Nagyová, Ingrid, 2014. *LEGO MINDSTORMS VE VYUCE PROGRAMOVANI V JAZYCE JAVA*. JTIE. 6, 2, s. 17–24. ISSN 1803537X. Dostupné z: http://jtie.upol.cz/cz/artkey/jti-201402-0003.php.

Nezvalová, Danuše, 2006. *Výukový proces* [online]. [cit. 11. 5. 2017]. Dostupné z: http://esfmoduly.upol.cz/texty/vyuk_proces.pdf.

Oroma, J. O. et al. 2012. ALGORITHM FIRST, SYNTAX LATER APPROACH FOR TEACHING PROGRAMMING TO NOVICES IN TANZANIAN UNIVERSITIES: A CASE OF TUMAINI UNIVERSITY. In *Proceedings of EDULEARN12 Conference*, ISBN 978-84-695-3491-5.

Papert, Symour, 1993. Mindstorms: Children, Computers, And Powerful Ideas. Imprint unknown, ISBN 0786723882.

Parker, Kevin R. et al., 2006. Criteria for the selection of a programming language for introductory courses. In *International Journal of Knowledge and Learning 2*. Dostupné z: https://www.researchgate.net/publication/220428331_Criteria_for_/the_selection_of_a_programming_language_for_introductory_courses

Pecinovský, Rudolf, 2013. *Metodologhy Architecture first* [online].[cit. 19. 3. 2017]. Dostupné z: http://vyuka.pecinovsky.cz/prispevky/2013_DIG_Metodika_Architecture_First.pdf.

Pecinovský, Rudolf, 2004. *BlueJ – vývojové prostředí pro výuku jazyka Java*,

Proctor, Chris et al., 2016. How We Teach Programming in Why We Teach Programming. In *Constructionism in Action 2016*, ISBN 978-616-92726-0-1.

Salanci, Lubomír, 2012. Vývoj didaktiky programovania. In *Didinfo 2012*, Dostupné z: http://didinfo.umb.sk/public/filestore/documents/richtext/138/didinfo2012.pdf.

Schubert, Singrid, Schwill, Andreas, 2011. *Didaktik der Informatik*. Spektrum-Akademischer Vlg, ISBN 978-3-8274-2652-9.

Skiena, Steve, 2008. *The Algorithm Design Manual*. Springer London Ltd. ISBN 978-1848000698.

Škavrda, Libor, 2006. *Co je to funkcionální programování* [online]. [cit. 5. 3. 2017] Dostupné z: http://programujte.com/clanek/2006032503-co-je-to-funkcionalni-programovani/.

Skoupil, David, 2007a. *Programy a projekty jazyku Scheme I.*

Skoupil, David, 2007b. *Úvod do paradigmat programování* [online]. [cit. 11. 5. 2017]. Dostupné z: https://phoenix.inf.upol.cz/esf/ucebni/uvod_para.pdf.

ŠPÚ, 2008. *Štátny vzdelavaci program* [online]. [cit. 22. 4. 2017]Dostupné z: http://www.statpedu.sk/sites/default/files/dokumenty/statny-vzdelavaci-program/informatika_isced3a.pdf

ŠPÚ, 2015. *Inovovaný štátny vzdelavaci program* [online]. [cit. 22. 4. 2017]. Dostupné z:http://www.statpedu.sk/sites/default/files/dokumenty/inovovany-statny-vzdelavaci-program/informatika_g_8_r.pdf

ŠPÚ, 2016. *Metodické usmernenie Č.1/2016* [online]. [cit. 22. 4. 2017]. Dostupné z: http://www.statpedu.sk/aktuality/metodicke-usmernenie-c12016

Štoffová, Veronika, Takáč, Ondřej 2013. ROBOTICKE STAVEBNICE V PRIPRAVE UCITELOV INFORMACNEJ VYCHOVY.In *Trends in education*. s. 315–322. ISSN 18058949. Dostupné z: http://tvv-journal.upol.cz/artkey/tvv-201301-0070.php.

Švec, Štefan, 1998. *Onsahová analýza textových dokumentů*. ISBN 80-88778-73-5.

TIOBE, 2017. *TIOBE:Latest news*. Dostupné z: https://www.tiobe.com/tiobe-index/

Writing@CSU, 2017. *Writing@CSU Guide: Content Analysis*, Dostupné z: https://writing.colostate.edu/guides/pdfs/guide61.pdf.

Zelle, John M., 1999. *Python as a First Language* [online]. [cit. 20. 6. 2017]. Dostupné z: http://mcsp.wartburg.edu/zelle/python/python-first.html.