Programovací jazyky a jejich využití

Programming languages and their usage

Bc. Michal Havránek

Diplomová práce
2011

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2010/2011

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Michal HAVRÁNEK**
Osobní číslo:	**A09501**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Programovací jazyky a jejich využití**

Zásady pro vypracování:

1. **Stručně seznamte s využívanými programovacími jazyky.**
2. **Popište metodiku empirického výzkumu.**
3. **Zpracujte studii dle sektorů.**
4. **Vyhodnoťte výsledky studie z pohledu odvětví.**

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **PUNCH, Keith F. Základy kvantitativního šetření. 1. vyd. Praha : Portál, 2008. 150 s. ISBN 978-80-7367-381-9.**
2. **Dotazník - Struktura dotazníku [online]. 2007. Dotaznik-online.cz, c2007 [cit. 2009-07-14]. Dostupný z WWW: [http://www.dotaznik-online.cz/zaklady-dotazniku.html].**
3. **Dotazník: Zásady tvorby dotazníku [online]. 2007 [cit. 2009-04-02]. Dostupný z WWW: [http://www.dotaznik-online.cz/index.html].**
4. **YOUNG, Stephen J. Programovací jazyky pro RT-aplikace. 1. vyd. Praha : Státní nakladatelství technické literatury, 1988. 338 s.**
5. **Most Popular Programming Languages. TIOBE.com [online]. 2010, n/a, [cit. 2011-01-24]. Dostupný z WWW: [http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html].**

Vedoucí diplomové práce: **Ing. Radek Šilhavý, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce: **24. února 2011**

Termín odevzdání diplomové práce: **18. května 2011**

Ve Zlíně dne 24. února 2011

L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

ABSTRAKT

Cílem této práce je popsat problematiku využití programovacích jazyků v rámci softwarových firem v české republice. Velké množství programovacích jazyků a souvisejících technologií vytváří otázky, na kterou technologii zaměřit svou pozornost ve výuce. Jsou rozebrány základní vlastnosti nejběžnějších programovacích jazyků. V části práce zabývající se empirickým výzkumem je zmíněn postup průzkumu trhu práce a dotazníkového šetření. V praktické části jsou vyhodnoceny výsledky z hlediska jednotlivých odvětví a jsou zde vykresleny různé vztahy a vazby vyplývající z výsledků práce. Ve zhodnocení výsledků je popsáno, jakým směrem se ubírá současný a budoucí technologický proces v oblasti programovacích jazyků a vývoje aplikací. Jsou doporučeny konkrétní oblasti, na které je potřeba se zaměřit ve výuce.

Klíčová slova:

Programovací jazyk, sektorový průzkum, empirický výzkum, dotazníkové šetření

ABSTRACT

The aim of this thesis is to describe the problems of programming languages in software companies in the Czech Republic. Due to a great number of programming languages and related technologies it is questionable on which technology to concentrate during classes. The thesis explains basic features of the most common programming languages. The empirical research in the thesis presents the steps taken during job market research and questionnaire survey. The practical part evaluates the results from the viewpoint of individual branches and describes various relations and connections ensuing from the results. Evaluation of the results describes the way the current and future technological process in the field of programming languages and application development is taking. The thesis recommends specific areas which require better focus during classes.

Keywords:

Programming language, sectional exploration, empirical research, questionnaire

Poděkování:

Děkuji svému vedoucímu diplomové práce Ing. Radku Šilhavému, Ph.D., za trpělivost, užitečné rady a doporučení na studijní materiály při zpracování mé diplomové práce.

Děkuji také panu Ing. Vítu Trnkovi z firmy GEPRO spol. s r.o. za velmi podnětné informace k tématu využití programovacích jazyků a rozsahu požadovaných znalostí u softwarových vývojářských firem.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl jsem seznámen s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně 2. května 2011 …….

podpis diplomanta

OBSAH

ÚVOD

Jako ve všech oblastech současného života se i oblast informačních technologií zaměřená na programování a využití programovacích jazyků neustále vyvíjí.

S nárůstem všech služeb, které jsou poskytovány lidem, vzrůstá poptávka po kvalitním softwaru. Co bývalo dříve dostupné jenom v podnikové sféře, tak se v současné době stává dostupné všem lidem. Domy jsou vybaveny inteligentními spotřebiči, automobily mají bohaté možnosti díky palubním počítačům a mnohé další vymoženosti nacházíme již zcela běžně.

Ale také průmysl dostává nové impulsy v podobě nových strojů napojených na centrální počítače, jsou zaváděny stále modernější ekonomické systémy, které pomáhají úspěšně vést podnik. Jsou zdokonalovány bankovní služby, ulehčující v mnohém práci a šetřící čas.

Na internetu jsou zaváděny služby, které poskytují bezpečné nákupy, je možné objednávat letenky a nepřeberné množství dalších služeb. Státní správa může díky novým technologiím lépe poskytovat servis široké veřejnosti.

Nic z výše jmenovaných věcí by prakticky nebylo uskutečnitelné bez kvalitního software. Vývoj software se odvíjí od znalostí programovacích jazyků a jim blízkým technologiím. Jak vyplývá z předchozího textu, vzrůstající technizace vyžaduje nové programátory.

Je tedy na místě si položit otázku, jaké programovací jazyky je vhodné začlenit jak do výuky na vysokých školách, tak do znalostí programátorů všeobecně. Není v lidských schopnostech naučit se opravdu detailně mnoho programovacích jazyků, protože každý jazyk vyžaduje na profesionální úrovni používání hluboké znalosti jak samotného programovacího jazyka, tak i souvisejících technologií. Rozhodnutí pro konkrétní typ programovacího jazyka se může stát základem pro několik let další práce programátora. Proto je třeba důkladně zvážit, v jakém oboru se chce potenciální programátor pohybovat a podle toho vhodně volit směr další výuky.

V této práci, která je zaměřena právě na výzkum používání programovacích jazyků, budou provedeny potřebné průzkumy trhu, aby se dala udělat ucelená představa o současném stavu. Zjištěné výsledky se v následujících několika letech můžou stát oporou při rozhodování, kterým směrem se ubírají požadavky na znalosti programovacích jazyků a kterým typům programovacích jazyků tedy věnovat pozornost při výuce.

I. TEORETICKÁ ČÁST

1 PROGRAMOVACÍ JAZYK

Budeme-li chtít, aby počítač zpracovával předložená data, je třeba mu říci, jakým způsobem to má udělat. Takový požadavek je počítači předložen v podobě tak zvaného programu. Počítačový program lze potom vytvořit pomocí programovacího jazyka.

Programovací jazyk je uměle vytvořený jazyk, který je používán k provádění výpočtů na strojích a počítačích. Díky němu lze kontrolovat chování stroje podobně jako je tomu při komunikaci mezi lidmi. Těchto programovacích jazyků je v současné době nevyčíslitelné množství. Některé jsou užívány pro přísně specifické úkoly a jiné typy mají mnohem širší záběr použití.

Programovací jazyky taky mají své standardy, jejichž dodržování je podmínkou pro správnost kódu a možnosti jeho úprav a začlenění do dalších celků.

Syntaxe programovacích jazyků je většinou textová, kde se jedná o slova, čísla a data, která jsou zapisována jako v běžně používaném jazyku. Z druhé strany ale existují i takové, které jako syntaxi používají grafických značek, které vyjadřují vizuální vztahy mezi symboly specifikujícími program. Snahou v obou případech je přiblížit se syntaxí takovému projevu, který by byl srozumitelný pro člověka.

Po syntaxi, která definuje pouze povrchovou strukturu jazyka, je tady ještě sémantika, která dotváří komplexnější popis. Sémantika říká, jaký efekt či chování bude následovat při vykonávání dané části kódu. Avšak není definována standardní cesta k tomuto popisu.[1]

```
if (x != 0) y = 1/x;                                    (1)
```

Příklad (1) ukazuje část kódu programovacího jazyka, který je zapsán pomocí daných pravidel syntaxe. Sémantika zde zase definuje, co nastane při splnění či nesplnění podmínky IF. Pakliže bude podmínka splněna, to znamená, že výraz bude vyhodnocen jako pravdivý, provede se výpočet y = 1/x. V opačném případě bude kód pokračovat dále.

Kritéria, podle kterých můžou být programovací jazyky rozeznávány, jsou různé. V následujících podkapitolách budou uvedena nejznámější kritéria.

1.1 Nízko úrovňové programovací jazyky

Do této skupiny programovacích jazyků patří všechny ty, které používají žádnou nebo minimální abstrakci nad systémem.

Jako abstrakci si lze představit stupeň těsnosti k hardwaru počítače. Dalo by se říci, že čím je stupeň abstrakce nižší, tím je syntaxe programovacího jazyka méně podobná běžnému textovému projevu lidí a tím pádem je i méně srozumitelná při čtení kódu.[1]

Programování bývá také náročnější na znalosti programátora. Setkáváme se s přímým přístupem k paměti a hardwarovým zařízením. Může zde být používáno přímo příkazů procesoru. Nízkoúrovňové programovací jazyky jsou používány mimo jiné k vytváření ovladačů zařízení.

Výhody tohoto typu programovacích jazyků jsou taky využívány u složitých výpočetních algoritmů, kde nízkoúrovňový programovací jazyk dokáže několikanásobně zrychlit provedení úkonu, protože lze programový kód algoritmu optimalizovat pro cílovou platformu.

Jako nejznámější lze vzpomenout jazyk symbolických adres (JSA). Taky je zde někdy řazen i programovací jazyk C, i když tento programovací jazyk je využíván i na vyšších úrovních abstrakce.

1.2 Programovací jazyky vyšší úrovně

Do této kategorie spadají programovací jazyky, které již pro přístup k hardwaru používají abstrakci. Tyto jazyky již nejsou závislé na strojových principech počítače. Je zde možnost přenositelnosti kódu na jiný systém.[2]

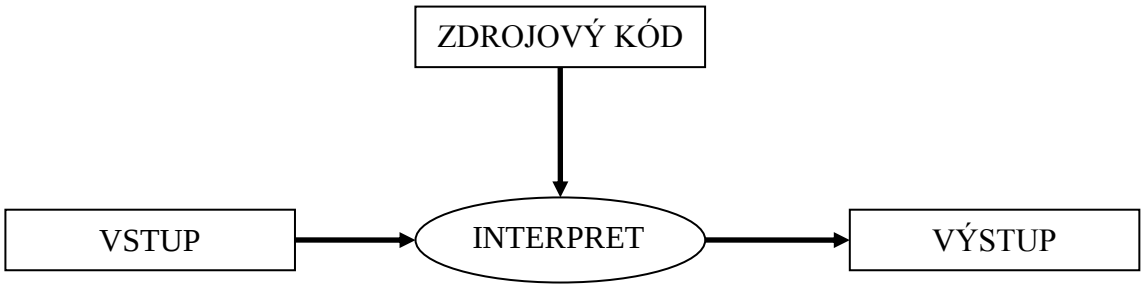
Díky menší těsnosti při práci s hardwarem nemusí být možné naprogramovat veškeré požadované akce. V takovém případě je nutné kód kombinovat s nízkoúrovňovým programovacím jazykem.[2]

Výhoda těchto typů programovacích jazyků je v jednodušším kódu a snadnějším vytváření aplikací. Do této kategorie spadá například programovací jazyk Java.

1.3 Programovací jazyky interpretované

Interpretované programovací jazyky potřebují ke svému spuštění tak zvaný interpret, kterým je kód překládaný za běhu samotného programu (Obrázek 1). Tento interpret musí být přítomen na cílovém stroji. Výhodou je to, že v takovém programu není nutné deklarovat proměnné či velikost paměti, kterou mají proměnné zabírat. Tento způsob programování může vytvořit jisté zjednodušující návyky pro programování aplikací.

Nevýhodou je, že takový program je potom zpomalen na své rychlosti, která závisí na tom, jak rychle je kód zpracován interpretem. Rychlost je ale oproti kompilovaným jazykům o poznání nižší.[2]



Obrázek 1 – Interpretovaný programovací jazyk

U některých programovacích jazyků této skupiny může interpret nejpoužívanější část kódu zkompilovat, aby tím urychlil pozdější spouštění dané aplikace. Zde se v poslední době projevuje především programovací jazyk Java a jeho překladač JIT (Just In Time).

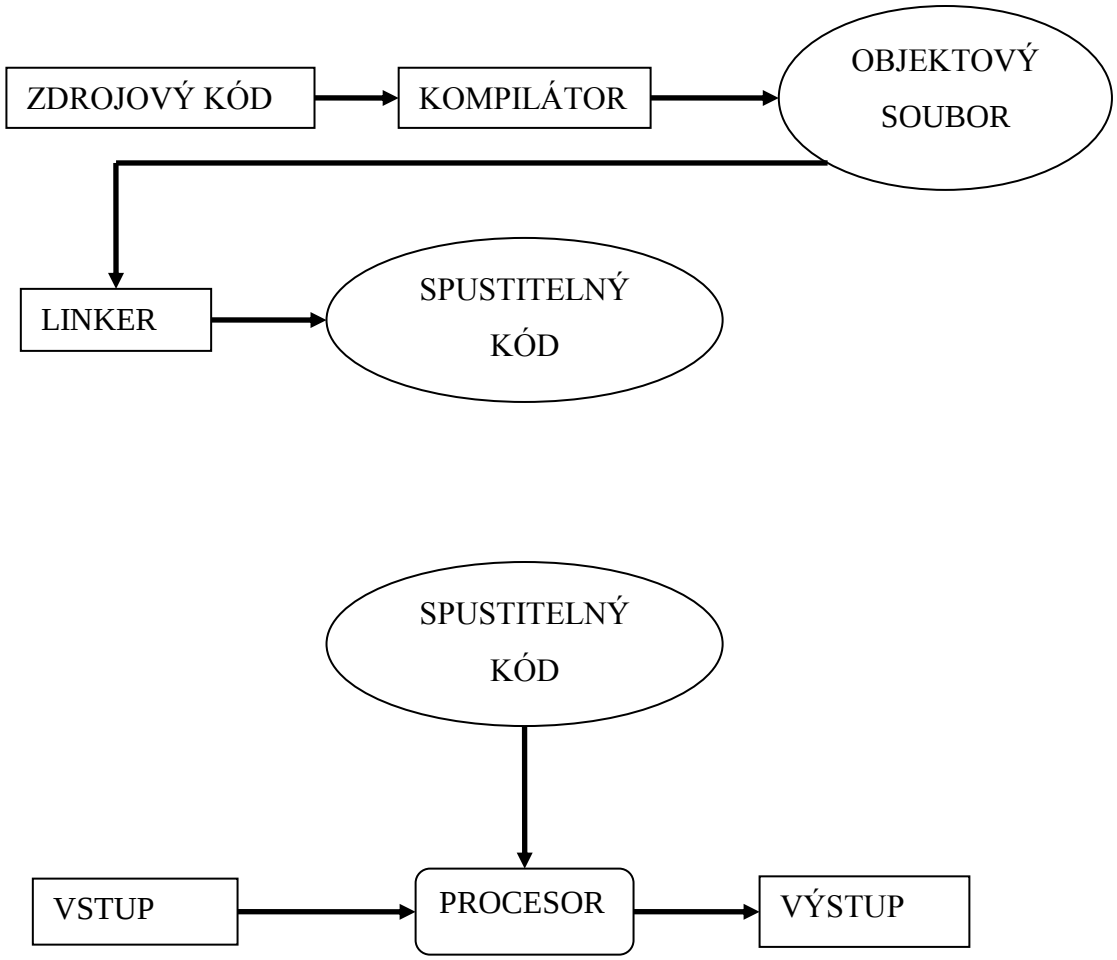
1.4 Programovací jazyky neinterpretované

Neinterpretované programovací jazyky (viz Obrázek 2) již nepotřebují ke svému spuštění na cílové platformě interpret. Výsledný program je již přímo spustitelný na cílové platformě. Proto musí být programový kód před svým použitím zkompilován do strojového kódu počítače.

Kompilátor slouží k převedení zdrojového souboru na program. Výsledkem kompilace je objektový soubor. Stále se však nejedná o spustitelný program. Ten je získán až po spuštění sestavovacího programu – linkeru. Linker provede spojení jednoho nebo více objektových souborů s potřebnými knihovnami. Knihovna je sbírka propojitelných souborů, které jsou buď součástí kompilátoru, nebo jsou dodány zvlášť.[3]

Pro každou platformu, pro kterou má být vytvořen spustitelný soubor je třeba volit odpovídající kompilátor a linker.

Mezi zástupce neinterpretovaných programovacích jazyků patří například C++.



Obrázek 2 – Neinterpretovaný programovací jazyk

1.5 Programovací jazyky imperativní

Tyto jazyky vystihují myšlenku koncepce počítače dle von Neumanna. Jsou charakterizovány sekvenčním vykonáváním příkazů, využíváním proměnných reprezentovaných v paměti a prováděním úkolů, které mění hodnoty proměnných. Někdy jsou tyto jazyky nazývány procedurální, avšak tento název nevystihuje přesně myšlenku procedury naznačené v úvodu kapitoly.

Imperativní jazyky ještě můžeme dělit [4]:

- Naivní – někdy se ani neuvádí, příkladem je jazyk BASIC

- Nestrukturované - velmi blízké jazykům symbolických adres. Programy jsou lineárními sekvencemi příkazů a skoky jsou v nich realizovány příkazem typu „goto“
- Strukturované - kvůli nepraktičnosti příkazu skoku „go to“. Jeho hlavním přínosem je fakt, že nahrazuje příkazy skoku pomocí podmíněných cyklů

V dnešní době je většina programovacích jazyků imperativní jako třeba jazyk C.

1.6 Programovací jazyky funkcionální

Programovací jazyk je již méně závislý na von Neumannově schématu. Každá sekvence příkazů je vykonávána na určité části dat. Je zde možnost paralelního výpočtu, který nemusí mít přesně dané pořadí.

Charakteristika těchto jazyků vychází z výpočtu hodnoty funkce nebo z volání funkce. Proto jsou někdy nazývány taky jako jazyky aplikační. V tomto smyslu se jedná o opak objektově orientovaných jazyků, protože funkcionální programování se soustředí raději na hodnoty a funkce než na oblast objektů v paměti.[1]

Tyto programovací jazyky vycházejí z matematického modelu lambda-kalkul. Lambda kalkul je teorie funkcí založená na velmi jednoduchém jazyce. Základní prvky tohoto jazyka (tzv. lambda výrazy) jsou pouze tři: proměnná, aplikace a abstrakce.[5]

- Proměnná reprezentuje blíže nespecifikovanou hodnotu.
- Aplikace představuje volání funkce s jedním argumentem
- Funkce v lambda kalkulu jsou reprezentovány ve formě abstrakce tvořené proměnnou označující parametr a tělem ve tvaru lambda výrazu.

Jako zástupce funkcionálně orientovaných jazyků je možné vzpomenout jazyk LISP.

1.7 Programovací jazyk logický

Tyto programovací jazyky jsou založeny na symbolické logice. Tyto jazyky bývají nazývány taky jako deklarativní, protože definujeme vlastnosti, ale nikoliv již prováděcí kód.[1]

V tomto typu jazyků je definována sada stavů, které definují, jaká operace má být provedena k dosažení požadovaného výsledku. Zástupce jazyků logických je například Prolog.

1.8 Programovací jazyk objektově orientovaný

Programátory byly znovu a znovu hledány řešení starých problémů. Což je protikladem stavu, kdy lze kód používat opakovaně. Myšlenka skrývající se za opakovanou použitelností kódu spočívá v tom, že vytvořené komponenty mají přesně definované vlastnosti a je možné je zabudovávat do programů dle potřeby. Objektově orientované programování se pokouší vyjít těmto požadavkům vstříc. Podstata spočívá ve vymodelování objektů spíše než dat.[3]

Základní pilíře objektového programování jsou 3.

1. Zapouzdření – objekt lze použít i bez toho, aby bylo známo nebo aby se uživatel staral o to, jak uvnitř funguje
2. Dědičnost – je možné deklarovat nový typ, který rozšiřuje již existující typ a přebírá jeho vlastnosti
3. Polymorfismus (Mnohotvarost) – znamená to, že stejné jméno může mít mnoho forem a podle požadavku se zvolí ta správná

Nejznámějším zástupcem objektově orientovaného programování je jazyk C++.

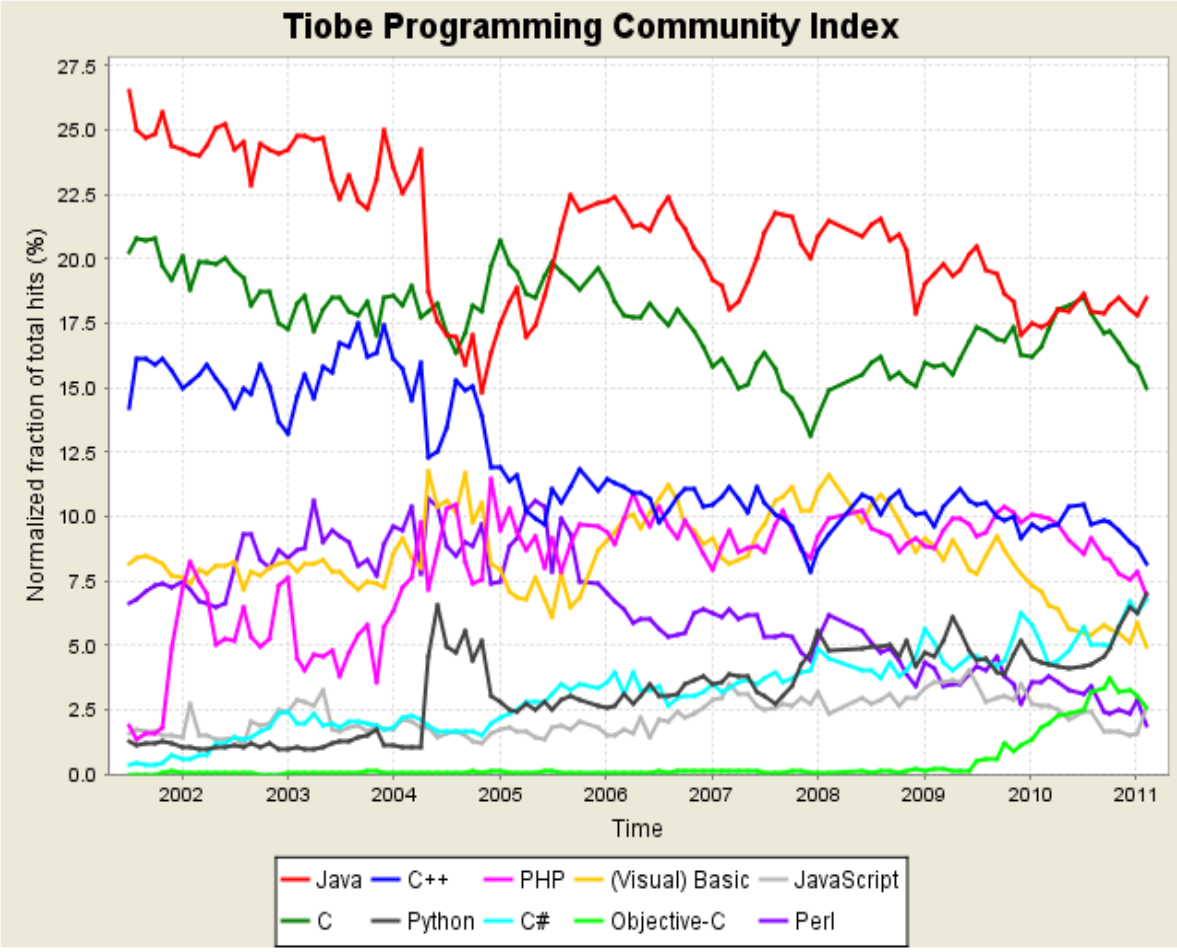
2 POPIS NEJPOUŽÍVANĚJŠÍCH PROGRAMOVACÍCH JAZYKŮ

V následujících kapitolách bude uveden popis a stručná historie několika nejznámějších a nejoblíbenějších programovacích jazyků. Toto rozdělení je převzato z celosvětového ohlasu uživatelů, který je zaznamenán za období od února 2010 do února 2011. Výzkum provedla společnost Tiobe Software.[6]

Position Feb 2011	Position Feb 2010	Delta in Position	Programming Language	Ratings Feb 2011	Delta Feb 2010	Status
1	1	=	Java	18.482%	+1.13%	A
2	2	=	C	14.986%	-1.62%	A
3	4	↑	C++	8.187%	-1.26%	A
4	7	↑↑↑	Python	7.038%	+2.72%	A
5	3	↓↓	PHP	6.973%	-3.03%	A
6	6	=	C#	6.809%	+1.79%	A
7	5	↓↓	(Visual) Basic	4.924%	-2.13%	A
8	12	↑↑↑↑	Objective-C	2.571%	+0.79%	A
9	10	↑	JavaScript	2.558%	-0.08%	A
10	8	↓↓	Perl	1.907%	-1.69%	A
11	11	=	Ruby	1.615%	-0.82%	A
12	-	=	Assembly*	1.269%	-	A-
13	9	↓↓↓	Delphi	1.060%	-1.60%	A
14	19	↑↑↑↑↑	Lisp	0.956%	+0.39%	A
15	37	↑↑↑↑↑↑↑↑↑↑	NXT-G	0.849%	+0.58%	A--
16	30	↑↑↑↑↑↑↑↑↑↑	Ada	0.805%	+0.44%	A--
17	17	=	Pascal	0.735%	+0.13%	A
18	21	↑↑↑	Lua	0.714%	+0.21%	A--
19	13	↓↓↓↓↓	Go	0.707%	-1.07%	A--
20	32	↑↑↑↑↑↑↑↑↑↑	RPG (OS/400)	0.626%	+0.27%	A--

Obrázek 3 – Celosvětová oblíbenost programovacích jazyků

Dalším ukazatelem, který tato společnost shromažďuje je vývoj oblíbenosti programovacích jazyků za poslední roky.



Obrázek 4 – Vývoj celosvětové oblíbenosti programovacích jazyků

Samotná oblíbenost programovacích jazyků mezi uživateli ale ještě neznamená, že stejné pořadí oblíbenosti je taky u softwarových firem a profesionálních programátorů. V praxi je totiž užíváno takových programovacích jazyků, které jsou pro daný účel nejvhodnější podle různých kritérií.

Může se jednat o potřebu rychlosti programu, kde jsou určité typy programovacích jazyků ve výhodě (viz Kapitola 1.1).

Jindy je zase třeba dbát požadavku na rychlost vývoje aplikace. Je zřejmé, že programovací jazyky, jež neobsahují příliš mnoho nízko úrovňových programátorských operací a spoléhají na bohaté knihovny (například programovací jazyk C# nebo Visual Basic.NET) mají rychlejší vývojový cyklus.

Další možností volby programovacího jazyka u softwarových vývojářů může být požadavek na multiplatformní aplikaci.

2.1 Programovací jazyk Java

Za člověka, který stál za podnětem k vytvoření Javy, lze považovat spoluzakladatele společnosti Sun Williama Nelsona Joye, který je známý pod jménem Bill Joy. V druhé polovině sedmdesátých let dvacátého století chtěl Bill Joy vytvořit jazyk, který by kombinoval nejlepší vlastnosti z jazyka C a jazyka MESA. Při přepisu operačního systému UNIX v osmdesátých letech pak poznal, že pro tuto práci není zrovna jazyk C++ vhodným kandidátem. Potřeboval jazyk, který by mu umožnil napsat program kratší a efektivnější.[7]

V lednu roku 1991 se společně sešli pánové Bill Joy, James Gosling, Mike Sheradin, Patrick Naughton, projektoví vedoucí Sunu, nad projektem který se jmenoval Stealth Project. Tento projekt měl za cíl vytvořit systém pro domácí spotřebiče. Gosling nejprve pro dosažení cíle úkolu používal jazyk C++, který však nebyl shledán plně vyhovujícím.[7]

Nakonec Gosling začal pracovat na novém jazyku, který bude platformě nezávislý a který bude plně vyhovovat cílům projektu. Tento jazyk byl pojmenován Oak (dub), podle stromu, který rostl pod okny kanceláře. Za přímého předchůdce jazyka Oak lze považovat jazyk C++. V dalších letech byla zjištěna existence jiného jazyka se stejným názvem (Oak), a tak bylo potřeba nalézt nově vzniknutému jazyku jiné jméno. Autoři vzpomínají, že nalezení nového názvu nebylo vůbec jednoduché a hledání jim zabralo mnoho hodin. Inspirace se pak dostavila náhle cestou do místního bufetu na kávu. Java v americkém slangu znamená právě kávu.[7]

Jak již bylo zmíněno, jazyk Java vychází z jazyka C++. Oproti němu je však syntaxe Javy o něco jednodušší. Taktéž odpadly konstrukce, které programátorům způsobovaly problémy. S výjimkou několika primitivních datových typů můžeme o Javě mluvit jako o objektovém jazyce. Objektový přístup s sebou přináší možnosti tvorby robustních aplikací.[7]

Další velkou výhodou Javy je její přenositelnost a skutečnost, že je multiplatformní. Java je jazykem interpretovaným (viz Kapitola 1.3). To znamená, že místo skutečného strojového kódu se při kompilaci vytváří pouze takzvaný mezikód (ByteCode), který je při spuštění aplikace zpracováván interpretem, který je zde nazýván virtuální stroj Javy (JVM - Java Virtual Machine). Program tedy stačí napsat pouze jednou a spouštět lze všude tam, kde je nainstalován příslušný virtuální stroj.

Tento přístup s sebou kromě výše zmíněných výhod bohužel nese to, že běh aplikace, která je napsaná v Javě, je značně pomalejší, než kdyby byla psána jako klasická nativní aplikace. Aby tato pomalost běhu aplikací nebyla tak znatelná, je používán tak zvaný „Just In Time Compiler" (JIT) v provedení pro jazyk Java. Jeho použití zrychluje běh aplikace tím způsobem, že je mezikód (ByteCode), nebo jeho části, překládány do strojového kódu procesoru. Touto technologií je značně urychlen běh aplikace, ale zase jsou kladeny vyšší nároky na paměť. Správa paměti je zabezpečena pomocí objektu garbage collector. Ten vyhledává již nepoužívané části paměti a uvolňuje je pro další použití.[7]

2.2 Programovací jazyk C

Procedurální, imperativní programovací jazyk C vytvořil v roce 1972 Dennis Ritchie v Bellových laboratořích. Jeho záměrem bylo napsat systémový jazyk pro systémové programátory. Konkrétně pro operační systém UNIX. Ten byl původně napsán v jazyce symbolických adres (JSA), ale od Kena Thompsona, jednoho z jeho vývojářů, vyvstal požadavek na jeho přepsání do některého z vyšších programovacích jazyků, protože JSA je závislý na určitém procesoru počítače. Byla zde tedy potřeba snadnější přenositelnosti systému UNIX mezi různými počítači. Nejprve byl zkoušen jazyk Fortran. Po několika dnech práce však byl Fortran shledán nepoužitelným. Thompson si následně napsal jazyk „B". Jazyk „B" byl jazykem zcela funkčním. Jednalo se však o jazyk interpretovaný (viz Kapitola 1.3) a tedy značně pomalý. Dennis Ritchie posléze do jazyka „B" implementoval datové typy a následně pak zahájil práci na kompilátoru pro tento jazyk. Výsledkem popsané činnosti byl na konci zcela nový jazyk C.[7]

Jazyk C je univerzální a flexibilní programovací jazyk, který není závislý na konkrétní hardwarové a softwarové platformě. Dlouho byl považován za systémový, neboť s jeho pomocí byly psány velké kusy operačních systémů, zejména UNIX. I když klasifikaci systémového nízkoúrovňového jazyka si jazyk C zachoval, je v současné době čím dál tím častěji používán k tvorbě všech typů programového vybavení. Denis Ritchie původně navrhl a implementoval jazyk pod operačním systémem UNIX. Podoba jazyka však byla časem vylepšována a normována a v současné době je v různých variantách vyráběn různými výrobci téměř na všech komerčně dostupných operačních systémech.[8]

Jazyk C má velmi úsporné vyjadřování, je strukturovaný, má velký soubor operátorů a moderní datové struktury. Pro mnoho úloh je efektivnější a rychlejší než jiné jazyky.

Není specializovaný na jednu oblast používání. Definujeme-li jazyk C jako nízkoúrovňový, znamená to, že C pracuje přímo pouze se standardními datovými typy, jako jsou znaky, celá čísla a reálná čísla. Jazyk C neumožňuje přímo práci s řetězci a poli ani přímo neobsahuje nástroje pro vstupy a výstupy. Tyto všechny funkce je nutné provádět pomocí volání funkcí, což přináší dvě základní výhody a to jednoduchost jazyka a jeho nezávislost na počítači.[9]

Z těchto výhod vyplývá:

- Snadné vytvoření překladače pro konkrétní počítač a konkrétní operační systém
- Velká efektivita kódu – program v C se téměř vyrovná programu v jazyce symbolických adres

Prvním standardem jazyka C se stala verze jeho autorů Dennis Ritchie a Brian Kernighan, která byla popsána v jejich knize „The C Programming Language“, která vyšla v roce 1978. Dlouhou dobu byl potom jako standard veden ANSI C, který právě z této knihy vycházel. Později byl tento standard ještě rozšířen.[9]

Jazyk C je přísně typový. V současné době je taky používán k vývoji úspěšného a spolehlivého operačního systému Linux.

2.3 Programovací jazyk C++

Autorem jazyka C++ je Bjarne Stroustrup z Bellových laboratoří. Původní jméno jazyka bylo „C with Classes“ (C s třídami) a až později byl jazyk přejmenován na C++ . Práce na jazyku C++ byly zahájeny na počátku osmdesátých let dvacátého století. Bellovy laboratoře pak jazyk C++ opustil v roce 1983.[7]

Jazyk C++ je tedy nadmnožinou jazyka C. Každý platný program v jazyce C je i platným programem v C++. Ty dvě znaménka „++“ právě značí jakousi přidanou hodnotu. Zatímco jazyk C je přísně strukturovaný jazyk, C++ vychází z objektově orientovaného přístupu (viz Kapitola 1.8). Jazyk C++ stejně jako jazyk C vyžaduje dodržování typů proměnných.

V roce 1995 vydal Bjarne Stroustrup knihu „The C++ Programming Language“ (Programovací jazyk C++) a došlo k prvnímu veřejnému uvolnění překladače jazyka C++. V roce 1998 byl pak jazyk C++ standardizován jak normou ANSI, tak i normou ISO. Norma je někdy souhrnně označována jako ANSI/ISO C++. Jazyk C je podmnožina jazyka

C++, avšak existují výjimky, kdy není možné některé programy v jazyce C překládat pomocí kompilátoru pro jazyk C++ .[7]

V programovacím jazyce C++ je kladen důraz místo na algoritmy na data. V jazyce C++ existuje slovo „class“, které popisuje specifikaci takovéto nové formy dat a „objekt“ potom představuje určitou datovou strukturu konstruovanou v souladu s daným plánem. Třída definuje všechny údaje, které slouží k reprezentaci objektu a činností, jež mohou být s těmito údaji prováděny.[10]

Při vytváření programu v C++ podle pravidel objektově orientovaného přístupu (viz Kapitola 1.8) nejprve navrhujeme třídy, které přesně popisují to, s čím program pracuje. Potom se pokračuje návrhem programu za použití objektů těchto tříd. Tím je podporováno vytváření znovupoužitelného kódu.[10]

2.4 Programovací jazyk Python

Python je moderní programovací jazyk, který vyvinul Guido van Rossum. Python má širokou paletu možností. Umožňuje vytvářet aplikace mnohem rychleji než při programování v tradičních jazycích, jako jsou C, C++ nebo Java. Jedná se o jazyk strukturovaný, objektově orientovaný a schopný dalších rozšíření.

Python je platformě nezávislý jazyk, který běží stejně na Windows, unixových operačních systémech či na počítačích Macintosh. Lze jej používat pro psaní malých aplikací nebo skriptů, stejně jako pro vývoj velkých softwarových projektů. Poskytuje přístup k velmi výkonnému a uživatelsky jednoduchému grafickému uživatelskému rozhraní. Navíc je jeho vývoj otevřený a je zdarma.[11]

Tradiční jazyky, jakými jsou například C, Java nebo Pascal, mají společné charakteristiky, které jim dávají podobný vzhled: silné typy, statické typy, složité cykly a nutnost zápisu dlouhého kódu pro vykonání relativně krátkých úloh. Oproti tomu Python obsahuje všechny známé konstrukce, jako jsou cykly, rozhodovací výrazy, pole a tak dále, ale použití mnoha z nich je v Pythonu snazší.[11]

První výhoda je, že řízení paměti je automatické. Není třeba se zabývat alokací a dealokací paměti a obávat se volných odkazů. Java je jediná z výše zmíněných jazyků, která to nabízí.

Druhou výhodou je skutečnost, že typy jsou spojeny s objekty, nikoliv s proměnnými. Proměnné může být přiřazena hodnota libovolného typu; seznam může obsahovat objekty mnoha různých typů. To znamená, že není třeba předem deklarovat typ proměnné. Této výhodě jazyka Python se blíží například jazyk Java a PHP.

Třetí výhodou je skutečnost, že operace v jazyce Python jsou prováděny na mnohem vyšší úrovni abstrakce. To je dáno principem jazyka Python a standardním kódem knihovny, která je součástí jeho distribuce. Program na stažení stránky z Internetu může být na dva nebo tři řádky.

Další výhoda tohoto jazyka je, že pravidla syntaxe jsou velmi jednoduchá.

Python umožňuje rychlou tvorbu aplikací. Není neobvyklé, aby programování stejné úlohy zabralo v Pythonu pouze pětinu času a kódu ve srovnání s programováním v jazyce C.[11]

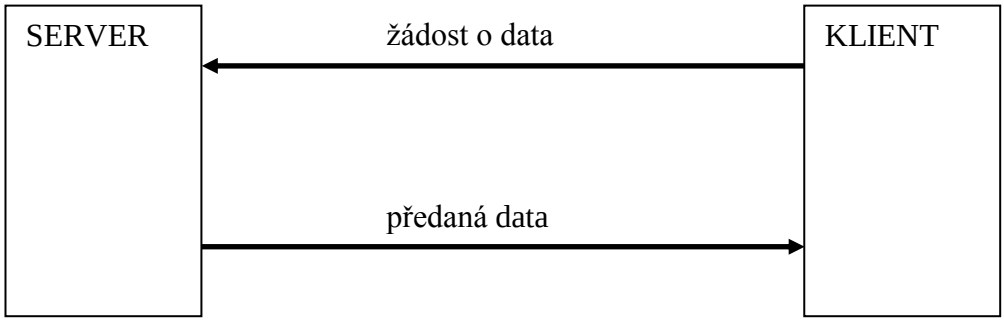
Určitou nevýhodou jazyka je rychlost provádění samotného programu, protože se nejedná o kompilovaný jazyk, ale je zde třeba mít na cílové platformě nainstalovaný interpret jazyka Python, podobně jako je tomu u jazyka Java.

2.5 Programovací jazyk PHP

PHP je stále mladý skriptovací programovací jazyk, původně navržený pro vytváření dynamických webových stránek. U počátku vzniku jazyka PHP je programátor dánsko-kanadského původu jménem Rasmus Lerdorf, který v roce 1994 řešil pro potřeby svých domovských stránek úlohu, jak vhodně získávat záznamy o přístupu čtenářů na svůj online životopis. PHP je jazyk platformě nezávislý.[7]

Někdy kolem roku 1995 vydal Lerdof sadu skriptů v jazyce Perl pod názvem PHP/FI. Tato sada skriptů se stala velmi používanou, a tak v roce 1997 vzniklo PHP/FI verze 2, které již bylo psáno v jazyce C. V roce 1997 se Andi Gutmans a Zeev Suraski pokoušeli použít PHP/FI pro vývoj komerčních aplikací a shledali jej jako poddimenzované pro tyto účely. Proto se rozhodli PHP/FI kompletně přepsat a výsledek byl označen jako PHP verze 3. V roce 1998 bylo přepsáno jádro PHP, aby bylo zvýšeno výkonu pro složité aplikace. Vzniklo tak PHP verze 4 s jádrem Zend podle počátečních písmen svých tvůrců. Následovala verze PHP 5, která vylepšovala možnosti objektově orientovaného programování.[12]

Jazyk PHP je založen na principu klient/server, to znamená, že k jeho běhu je potřeba HTTP server. Jedna strana je server, na kterém běží PHP a který na základě programu napsaném v PHP vytvoří požadovanou odpověď na dotaz, který je zaslaný klientem (viz Obrázek 5).



Obrázek 5 – Architektura klient/server

Při používání statických webových stránek není možné měnit jejich obsah jiným způsobem než přímou změnou zdrojového kódu stránky. U dynamických webových stránek je možná změna obsahu a na rozdíl od stránek statických se můžou zobrazovat pokaždé jinak.

Aby bylo možné provozovat dynamické webové stránky, musí být použito tak zvaného skriptovacího jazyka, což je v podstatě programovací jazyk, který se podílí na vytvoření webové stránky nebo mění její obsah. Dynamiku do webové stránky tedy přidává skriptovací jazyk. Existují dva druhy skriptovacích jazyků. Jsou to jednak skriptovací jazyky o které se stará webový prohlížeč (tedy klient) – ty se nazývají klientské skriptovací jazyky. Serverové skriptovací jazyky, mezi které právě patří PHP, běží na straně serveru a oproti klientským se používají k tvoření plnohodnotných aplikací, které spolupracují s databázemi, poštou a tak dále.[12]

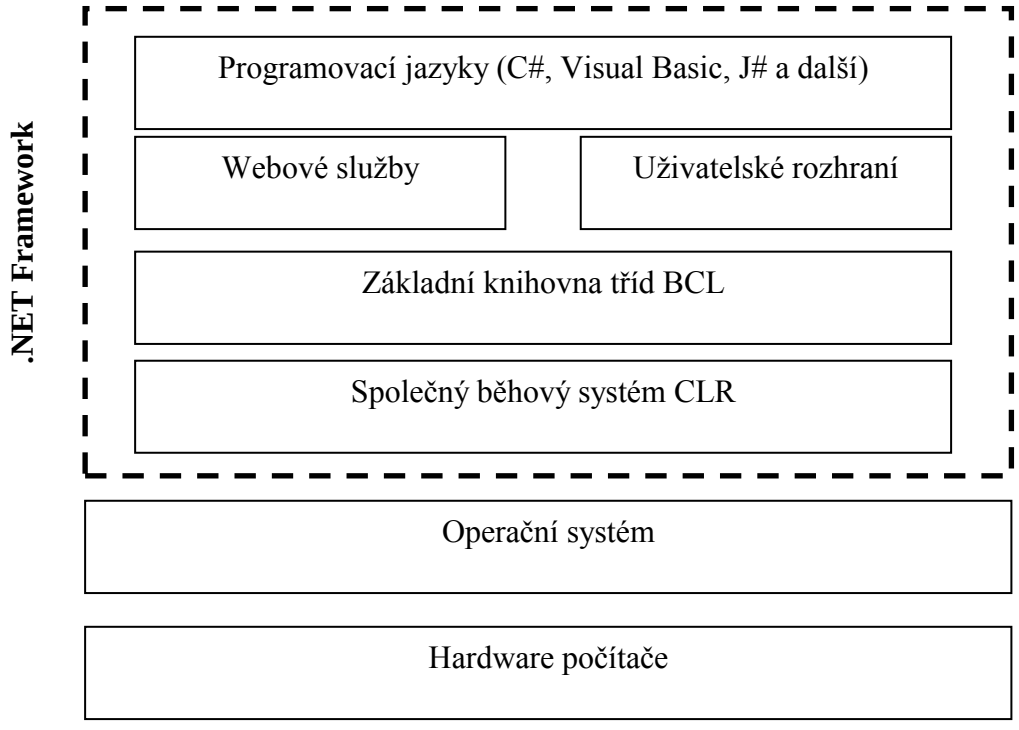
2.6 Programovací jazyk C#

Programovací jazyk C# (vyslovováno jako „sí šarp“) je žhavou novinkou počátku třetího tisíciletí.

Základní údaje o tomto jazyku jsou [13]:

- Tento jazyk vytvořila a šíří velká počítačová firma – Microsoft.
- Jde o jazyk určený pro tvorbu programů pro operační systém MS Windows.
- Tým, který jazyk C# navrhl, vedl Andres Hejsberg, autor programovacího jazyka Turbo Pascal a vedoucí týmu, který vyvinul Borland Delphi – dva velice úspěšné programovací jazyky.
- Návrh jazyka C# vychází ze zkušeností s jazykem C++, který byl nejpopulárnějším programovacím jazykem v devadesátých letech minulého století, a s jazykem Java, který se těší velké popularitě v současnosti.

Zdrojový kód programu napsaného v jazyce C# je přeložen nikoliv do strojového kódu počítače jako u neinterpretovaných jazyků (viz Kapitola 1.4), ale do univerzálního pomocného jazyka označovaného „Microsoft Intermediate Language“ (MSIL nebo jen IL). V poslední době je tento pomocný jazyk taky označován jako „Common IL“ (CIL) na znamení, že prostředí .NET Framework (Obrázek 6), které je nutné pro běh aplikace napsané v jazyce C#, bylo přijato jako standard organizací ECMA (European Computer Manufacturers Association).[13]



Obrázek 6 – Struktura prostředí .NET Framework

Pomocný jazyk je převeden do strojového kódu počítače zpravidla až v okamžiku, kdy program spustíme. To znamená, že na cílovém počítači musí být překladač, který překládá z IL do strojového kódu. Tento překladač se nazývá JIT (popis principu viz Kapitola 2.1). Pro běh aplikace musí být na cílovém počítači nainstalováno prostředí .NET Framework, které kromě překladačů JIT obsahuje i další součásti potřebné pro běh programů vytvořených v jazyce C#.[13]

Aplikace napsané v jazyce C# je v dnešní době možné spustit i na operačním systému Linux nebo třeba Mac OS X. Pro tyto účely je vyvíjena volně dostupná multiplatformní varianta .NET Frameworku, zvaná Mono. Kompatibilita sice není stoprocentní, ale s postupným vydáváním nových verzí se zlepšuje.

2.7 Programovací jazyk Visual Basic

Úplně v počátcích existoval vysokoúrovňový jazyk Basic, který byl určen k výuce programování začátečníků.

Společnost Microsoft potřebovala vhodný programovací jazyk, kterým by bylo jednodušší a rychlejší vytvářet grafické uživatelské programy pro jejich nový operační systém Windows. Dříve byla společnost Microsoft nucena vytvářet grafické uživatelské programy v jazyce C++ a to bylo nákladné a obtížné vzhledem k častému výskytu chyb. Bylo třeba vytvořit programovací jazyk, který bude pracovat na vyšší úrovni abstrakce.[14]

V roce 1991 byl tedy vyvinut jazyk Visual Basic (odvozeno od slova visual – vizuální). Jednalo se o událostmi řízený programovací jazyk.

Jazyk Visual Basic je relativně snadný na naučení. Jeho podstata při programování spočívá v tom, že vývojové prostředí, kterým je program vytvářen, obsahuje předdefinované visuální prvky připravené k okamžitému použití. Těmito prvky jsou jak jednotlivé komponenty grafického okna, tak samotné formuláře, na které jsou tyto prvky umisťovány. Samozřejmostí jsou implementované události těchto prvků a formulářů, na které stačí pouze napsat patřičnou funkci, která bude provedena při vyvolání dané události.

Samotný program napsaný v jazyce Visual Basic potřebuje ke svému spuštění speciální knihovnu, která musí být na cílovém počítači přítomná.

Postupně bylo vytvořeno několik verzí tohoto programovacího jazyka. Poslední verze měla číslo 6 a byla vypuštěna v roce 1998. Ale v roce 2002 se Microsoft rozhodl nevydávat další

novou verzi jazyka Visual Basic, ale bylo provedeno kompletní přepracování a přepsání tohoto jazyka. Nově vydaná verze byla pojmenována Visual Basic .NET 1.0. V následujících rocích byly opět vydávány novější verze.[14]

Verze jazyka Visual Basic .NET je závislá na platformě .NET Framework. Z toho důvodu nebylo možné zachovat kompatibilitu s předchozími verzemi programovacího jazyka Visual Basic. Programovací jazyk Visual Basic .NET je objektově orientovaný jazyk a již není přímo kompilován do spustitelného kódu, ale podobně jako jazyk C# do mezikódu jazyka MSIL (viz Kapitola 2.6).

Programovací jazyky Visual Basic a Visual Basic .NET jsou schopny vytvářet spustitelné aplikace pro operační systém Windows, dynamické knihovny, webové aplikace, ovládací prvky ActiveX , rozhraní k databázovým systémům, případně aplikace běžící jako služba operačního systému.

Všechny verze programovacího jazyku Visual Basic a Visual Basic .NET jsou určeny pro platformu Microsoft Windows.

2.8 Programovací jazyk Objective-C

Jazyk Objective-C vytvořil počátkem 80. let minulého století Brad Cox jako objektové rozšíření jazyka C a toto rozšíření bylo založeno na jazyku Smalltalk-80. Zůstává zachována rychlost a jednoduchost jazyka C.

V roce 1988 licencovala jazyk Objective-C společnost NeXT Software, která vyvinula pro tento jazyk knihovnu zvanou NeXTSTEP. V roce 1996 byla společnost NeXT Software koupena společností Apple. Následně provedla společnost Apple začlenění knihovny NeXTSTEP do jádra svého systému MAC OS X. Díky tomu dostal Apple moderní základ pro svůj operační systém.[15]

V dalším operačním systému od společnosti Apple s názvem iPhone nebo taky iOS se většina knihovny NeXTSTEP včetně jejich moderních optimalizačních vlastností objevuje také. V případě iOS se totiž jedná o redukovanou verzi operačního systému OS X.[15]

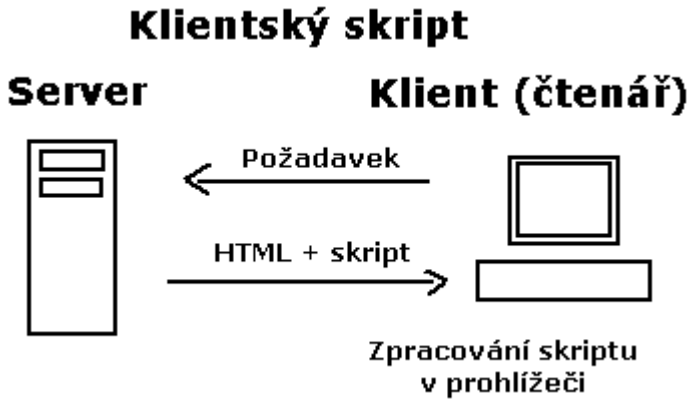
Apple přidal do jazyka Objective-C noho dalších vlastností, které rozšiřují jeho funkcionalitu o možnosti paralelního běhu, které jsou u ostatních jazyků teprve v počátcích. Tento hlavní přídavek do jazyka byl nazván jako Objective-C 2.0 a stále

zůstává jazykem, který lze používat jak v operačním systému OS X tak i v operačním systému iOS.[15]

2.9 Programovací jazyk JavaScript

JavaScript je programovací jazyk, který se používá v internetových stránkách. Kód se zapisuje přímo do HTML kódu stránky. JavaScript nemá nic společné s programovacím jazykem Java.

JavaScript je klientský skript. To znamená, že se program odesílá se stránkou do klienta (do prohlížeče) a teprve tam je vykonáván. Protikladem klientských skriptů jsou skripty serverové, které jsou vykonávány na serveru a na klienta jdou už jen výsledky (viz Kapitola 2.5).[16]



Obrázek 7 – Princip funkce JavaScriptu

Samotný jazyk vznikl počátkem roku 1996, kdy společnost Netscape vydala druhou verzi svého prohlížeče, která obsahovala mnoho nových funkcí. Jedna z nich byl JavaScript. Autorem byl Brendan Eich, který tento jazyk implementoval do webových stránek a byl schopný měnit a získávat data z formulářů. Ten samý rok byl JavaScript standardizován organizací ECMA. Postupně byl jazyk vylepšován a byly vydávány novější verze.

2.10 Programovací jazyk Perl

Jazyk Perl (Practical Extracting and Reporting Language) je interpretovaný programovací jazyk vyšší úrovně. Pracuje se s logickou syntaxí technického jazyka, jako jsou proměnné a přiřazení. Současná aktuální verze je již od roku 1994, proto se připravuje novější, která bude přepracována za účelem zvýšení rychlosti a efektivity jazyka.[17]

Perl byl kompletně vytvořen v jazyku C a i jeho chystaná novější verze bude psána v tomto jazyku z důvodu přenositelnosti a rozšiřitelnosti na různé platformy. Programy v jazyce Perl jsou pomalejší než programy v jazyce C, ale rychlejší v provádění skriptů než jazyk PHP.[17]

Perl může spolupracovat s ostatními jazyky, kdy je často využívána kombinace jazyka C a jazyka Perl.

Programy v jazyce Perl můžou mít dvě formy.

První forma je skript. V takovém případě je k dispozici zdrojový kód, kde je v prvním řádku uvedena cesta k interpretu. Jakmile je soubor spuštěn, interpret vykoná postupně všechny příkazy, které jsou v souboru zapsány. Výhoda je malá velikost souboru. Nevýhoda je volný zdrojový kód a pomalejší vykonávání instrukcí závislých na rychlosti interpreta.[17]

Druhá forma je binární forma. V takovém případě je soubor zkompilován pomocí vhodného překladače. Výsledkem je binární soubor, který obsahuje všechny potřebné knihovny a může být šířen na jakýkoliv systém. Zdrojový kód zůstává skrytý a program funguje nezávisle na přítomnosti interpretu na cílovém počítači. Nevýhoda je mnohem větší velikost souboru oproti velikosti ve formě skriptu.[17]

2.11 Programovací jazyk Ruby

Autorem tohoto mocného, dynamického, interpretovaného, plně objektově orientovaného skriptovací jazyka je Yukihiro Matsumoto. Tento autor započal vývoj tohoto jazyka v únoru 1993. První program pak napsal tentýž rok v létě.[7]

Interpret jazyka Ruby je napsán v jazyce C, což zaručuje jeho běh na nejrůznějších platformách. Ruby je vyvíjen jako open source a to znamená, že se jedná o otevřený zdrojový kód.

Návrh jazyka je nejvíce ovlivněn jazyky Perl, Smalltalk, Eiffel, Ada a Lisp a cílem bylo dosáhnout vyváženosti mezi funkcionálním programováním a imperativním programováním (viz Kapitoly 1.5 a 1.6).[18]

Na počátku vývoje byla taky snaha vytvořit takový skriptovací jazyk, který by byl mnohem silnější než jazyk Perl a mnohem více objektově orientovaný než jazyk Python. Dále bylo

snahou vytvořit počítačový jazyk, v němž se programátor může soustředit víc na řešení daného problému než na syntaxi jazyka samotného. Tedy vytvořit jazyk vyšší úrovně. V praxi si lze použití jazyka Ruby představit při řešení následujících typů úloh: zpracování textu, psaní CGI skriptů, síťové aplikace, tvorba grafického uživatelského rozhraní, práce s XML nebo třeba výuka objektově orientovaného programování.[7]

Ruby obsahuje ochranu proti výjimkám podobnou jako jazyky Java nebo Python. Součástí je také garbage collector pro automatické prohledávání a uvolňování nepoužívaných objektů. Tento jazyk lze snadno používat pro vytváření rozšíření pro jazyk C, knihovny můžou být načítány dynamicky při běhu programu. U programovacího jazyka Ruby může být používán multithreading (vícevláknové aplikace) bez ohledu na to, jestli daný cílový operační systém multithreading podporuje.

Ruby je plně přenositelný na většinu existujících cílových platforem.

2.12 Jazyk symbolických adres Assembler

Je nazývaný také jako jazyk symbolických instrukcí. Označení Assembler není úplně přesné, ale je standardně používáno širokou programátorskou veřejností a odbornými knihami. Proto je pod tímto názvem tento programovací jazyk mnohem známější. Jedná se o programovacím jazyk procesoru. To znamená, že není přenositelný na jinou platformu. Každý procesor nebo určitá řada procesorů má svůj vlastní jazyk symbolických adres. Z tohoto důvodu je jazyk velmi rychlý, protože pracuje přímo s procesorovými instrukcemi, je to jazyk nejnižší úrovně avšak poněkud složitější na běžné používání a vývoj aplikace trvá mnohem delší dobu než u vyšších programovacích jazyků. Kód tohoto programovacího jazyka je překládán do spustitelné podoby překladačem, který se jmenuje assembler.

Jazyk symbolických adres je vytvořen jako sada krátkých instrukcí, které říkají přímo procesoru, co má udělat. Tyto krátké instrukce jsou odvozeny od anglických zkratek slov, které vyjadřují danou operaci.

Programovací jazyky založené na používání symbolických adres se objevily v padesátých letech 20. století. Byly označovány jako nástupce přímého strojového kódu. Výhoda byla v tom, že už nebylo třeba si pamatovat číselné instrukce, které se používaly u přímého strojového kódu. S nástupem dalších programovacích jazyků bylo postupně používání

jazyka symbolických adres omezováno. Problémem taky byla závislost na daném typu procesoru.

Jazyk symbolických adres je v současnosti stále používán. I když byl v mnoha úsecích nahrazen vyššími programovacími jazyky, stále plní svou funkci v oblastech, kde je třeba využívat schopností přímého přístupu k instrukcím procesoru. Nezastupitelnou úlohu plní při programování ovladačů hardwarových zařízení, které potřebují přímý přístup do registrů svých pamětí. Jsou využívány v oblastech, kde je vyžadován vysoký výpočetní výkon. Své uplatnění také mají u systémů pracujících v reálném čase.

Po jazyce symbolických instrukcí následovaly jazyky jako Fortran, který byl snadno naučitelný a ALGOL, který se však příliš neujal. Na principu jazyka ALGOL vznikly jiné jazyky, z nichž nejpozoruhodnější je RTL/2. RT-jazykem se rozumí jazyk vhodný pro programování úloh probíhající v reálném čase. RTL/2 je jazyk nepříliš rozsáhlý, kompaktní a efektivní a je přitom vybaven řadou dobře koncipovaných mechanismů s velkou mírou zabezpečenosti, umožňujících přehledné vyjádření řídících struktur. Avšak zatímco RTL/2 obsahuje dobré možnosti abstrakce řízení, jeho možnosti abstrakce dat jsou nedostatečné.[19]

3 METODIKY VÝZKUMU

Platí, že ne všechny šetření mají kvantitativní charakter. Kvalitativní šetření, v kterých jsou obvykle pokládány otevřené otázky, neprodukují kvantitativní nebo numerická data. Lidé odpovídají na otevřené otázky celými větami a výzkumníci poté přistupují k analýze těchto vět, aniž by tyto věty byly transformovány do číselných hodnot.[20]

V této práci bude použito metodiky s kvantitativními daty.

Podstata kvantitativního výzkumu spočívá ve zkoumání vztahů mezi proměnnými. V kvantitativním výzkumu je realita zachycována pomocí proměnných, jejichž hodnoty jsou zjišťovány měřením a primárním cílem je nalézt, jak jsou proměnné rozloženy a zvláště jaké jsou mezi nimi vztahy.

Kvantitativní znamená, že šetření je navrženo tak, aby přineslo numerická data měřením proměnných.[20]

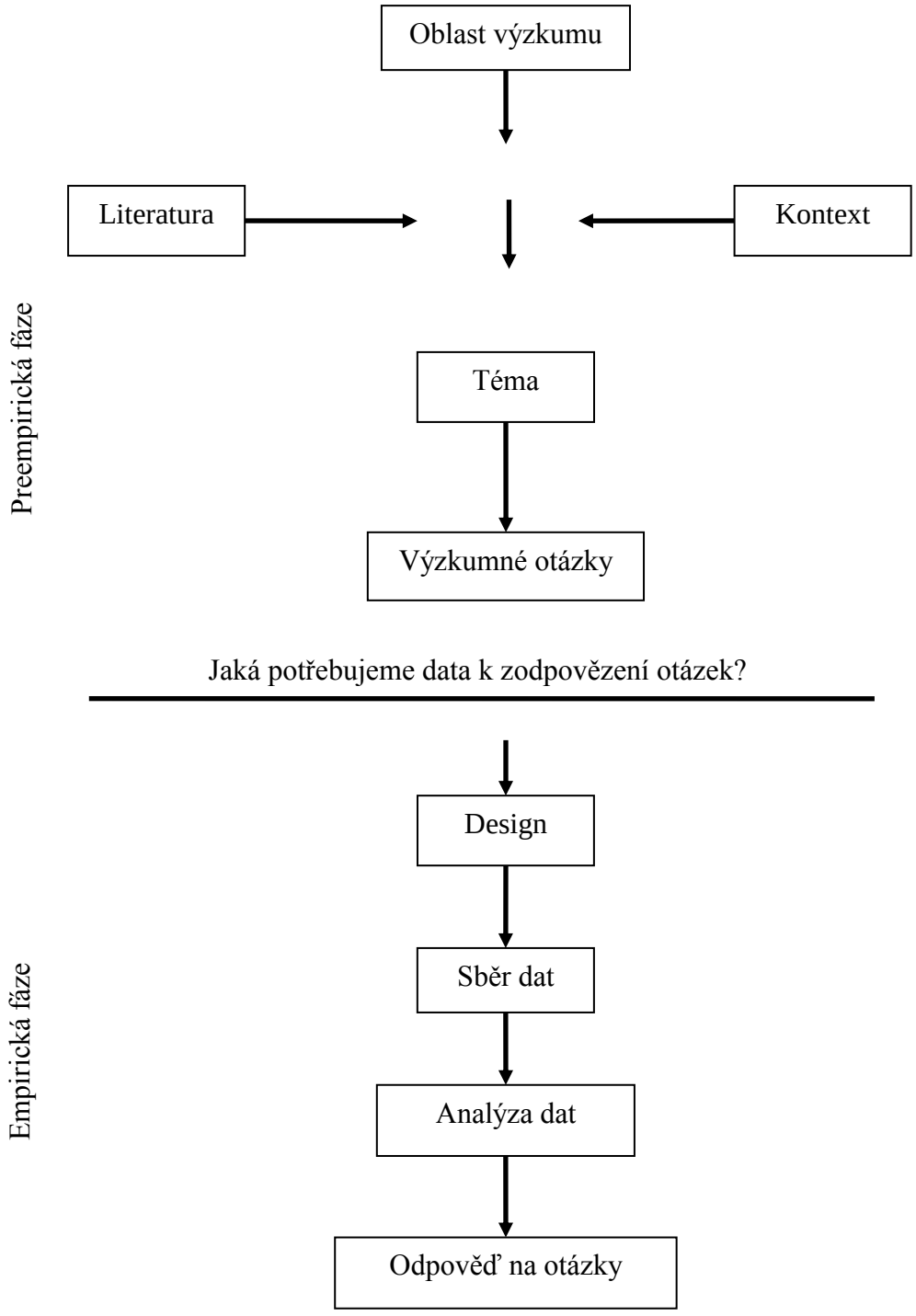
- Šetření malého rozsahu znamená, pozornost je věnována situacím s limitovanými zdroji pro výzkumníky.
- Průřezové šetření znamená, že jsou shromažďována data v jednom časovém okamžiku.
- Individuální osoba jako jednotka analýzy znamená, jak se liší jedinec na hodnotách proměnných od jiného jedince.

Nejběžnější metoda šetření u kvantitativního výzkumu je založena na dotaznících vyplňovaných respondenty.[20]

3.1 Empirický výzkum

Pojem empirický znamená, že je založený na zkušenostech. Vědy, které jsou založeny na empirických metodách, mohou být mimo jiné nazývány vědami empirickými.

Empirický výzkum je vhodný jak pro kvantitativní, tak pro kvalitativní studie. Na výzkum je pohlíženo jako na organizovaný, systematický a logický proces zkoumání používající empirické informace k zodpovězení otázek nebo k testování hypotéz.[20]



Obrázek 8 – Zjednodušený model výzkumu

Na obrázku (Obrázek 8) je znázorněn zjednodušený model výzkumu, který zdůrazňuje ústřední roli výzkumné otázky a má čtyři hlavní vlastnosti [20]:

1. Vymezuje výzkum pomocí výzkumných otázek

2. Určuje, jaká jsou zapotřebí data k zodpovězení těchto otázek
3. Navrhuje výzkumné cesty k získání a analýze dat
4. Používá data k zodpovězení otázek

Empirický výzkum může být definován i jako výzkum založený na experimentu a pozorování, který má otestovat naši hypotézu.

3.1.1 Druhy empirického výzkumu

Základní význam má rozlišení na primární a sekundární empirický výzkum.[21]

- Primární výzkum – zahrnuje vlastní zjištění hodnot vlastností u samostatných jednotek. Jedná se o tak zvaný sběr dat přímo v terénu.
- Sekundární výzkum – znamená dodatečné využití dřívějších dat, které již byly shromážděny někým jiným. Tato data jsou dále rozdělena na:
    - Data neagregovaná – původní podoba zjištěných dat
    - Data agregovaná – kdy jsou hodnoty vlastností sumarizované za celý soubor

Význam sekundárních agregovaných dat by mohl být spatřován v následujících třech rovinách [21]:

1. V případě statistických údajů mohou poskytnout již v přípravné fázi výzkumu vstupní informace o sledovaném problému
2. Bez těchto dat se neobejde výběr zkoumaného vzorku včetně závěrečného vyhodnocení reprezentativity.
3. V konečné fázi lze vypočítat hodnoty sledovaných problémů zjištěných na výběrovém souboru, za celou populaci, celý soubor.

V této práci bude využito právě agregovaných dat (viz Kapitola 2), která obsahují všeobecný přehled nejoblíbenějších programovacích jazyků a na základě kterých byl sestaven dotazník u této metody průzkumu.

3.2 Postup při výzkumu

Každý konkrétní výzkum se vyznačuje zvláštnostmi, které vyplývají z jedinečnosti povahy řešení. Obecně může být postup přiblížen jako proces sestávající z následujících pěti kroků [21]:

1. Definování problému a cílů výzkumu
2. Sestavení plánu výzkumu
3. Shromáždění informací
4. Statistické zpracování a analýza informací
5. Prezentace výsledků včetně praktických doporučení

3.2.1 Problémy a cíle empirického výzkumu

Formulace problému je jednou z nejdůležitějších částí výzkumu. Tematicky, obsahově vymezuje oblast, na kterou budou zaměřeny kroky výzkumu. Důležité je mít přesně definovaný problém, aby výsledky odpovídaly zadání. Je proto důležité zpracovat stručně teoretické vymezení problému, kde nezbytnou součástí je vymezení hlavních pojmů.[21]

3.2.2 Sestavení plánu

Zde jsou definovány potřebné informace a stanoveny postupy pro jejich získání, vyhodnocení a interpretaci.

3.2.3 Shromažďování informací

V tomto bodě jsou roztříděna data podle nejrůznějších kritérií. Je požadována relevantnost, validita, efektivnost.

3.2.4 Analýza situace

Zde je výzkumný pracovník seznamován s prostředím a podstatou problému. Je prováděno studium již existujících informací, konzultace s odborníky a jsou hledána data, která by mohla přispět k řešení problému.[21]

3.2.5 Prezentace výsledků

Probíhá závěrečné zpracování výsledků do projektu a jejich odevzdání zadavateli.

3.3 Techniky empirického výzkumu

V případě technik empirického výzkumu se jedná o konkrétní způsob sběru primárních dat, umožňujících evidovat výskyt jevů či chování, ale taky zjistit názory, postoje a motivy. Základní tři techniky výzkumu představují [21]:

1. Dotazování
2. Pozorování
3. Experiment

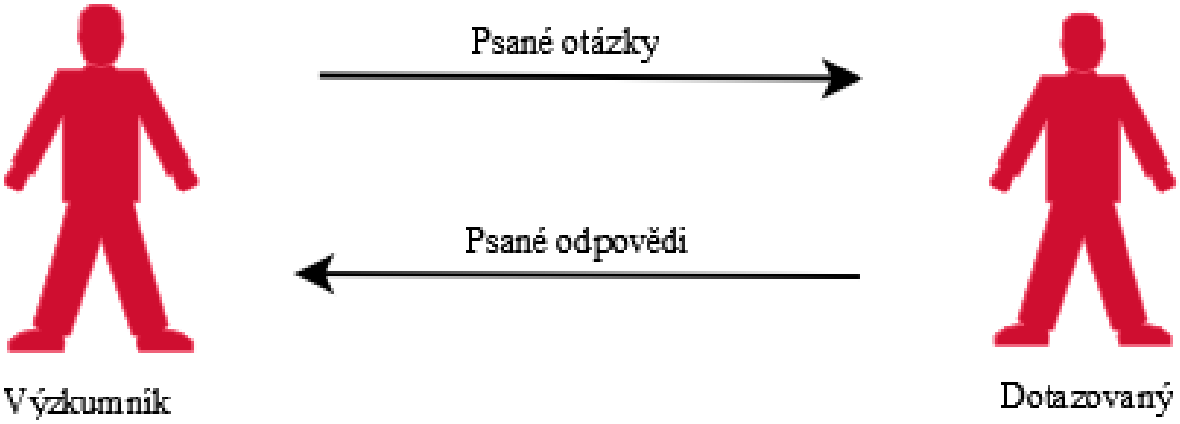
V této diplomové práci bude použito prvních dvou technik. V případě dotazování bude provedeno šetření v konkrétních softwarových firmách, které se zabývají vývojem software. U pozorování se bude jednat o sběr dat na úrovních pracovních nabídek, kterými softwarové firmy hledají nové zaměstnance na pozice programátorů.

3.4 Dotazování

Jak již bylo zmíněno v kapitole (Kapitola 3) k nejrozšířenějším postupům patří dotazování. Dotazování se uskutečňuje pomocí různých nástrojů (dotazníky, záznamové archy, osobní dotazování) a vhodně zvolené komunikace. Tento kontakt může být přímý, bezprostřední jako je tomu v případě písemného zodpovídání otázek. Dotazovaný je zde nositelem informace.[21]

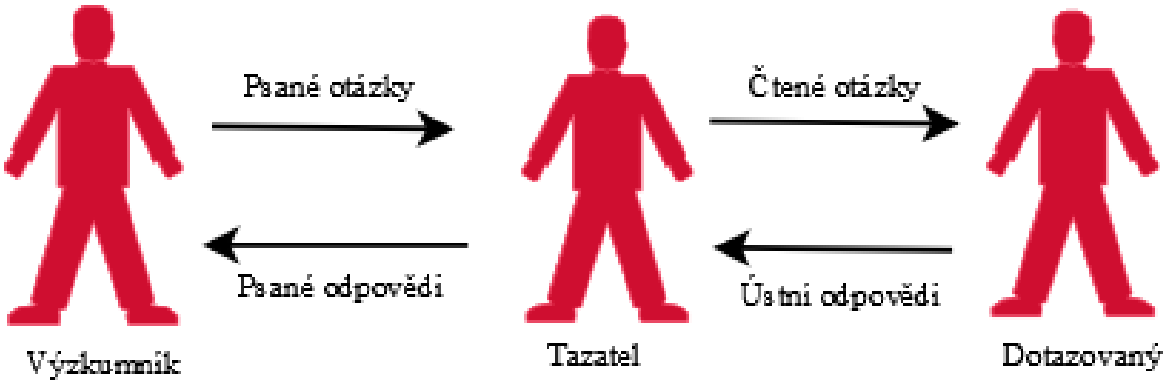
Tohoto způsobu bude využito v této diplomové práci. Pomocí dotazníkového šetření budou osloveny softwarové firmy a jejich odborníci, kteří svými postřehy a odpověďmi na položené otázky vytvoří patřičný soubor dat, ze kterého lze vyvodit požadované výsledky pro závěrečné zhodnocení.

Princip přímého dotazování je znázorněn na obrázku (viz Obrázek 9) a v dotazníkovém šetření této práce bude použito právě tohoto způsobu.



Obrázek 9 – Přímé dotazování

Druhou možností je zprostředkování pomocí tazatele, který vstupuje mezi výzkumníka a dotazovaného (viz Obrázek 10).



Obrázek 10 – Zprostředkované dotazování

V prvním případě (viz Obrázek 9) chybí zprostředkující osoba – tazatel, kterou by se měl dotazník maximálně snažit nahradit. Obsah dotazníku musí respondentovi sdělit vše podstatné. Naopak v druhém případě vstupuje mezi výzkumníka a dotazovaného vyškolený a instruovaný tazatel, kterému stačí záznamový list.[21]

3.4.1 Empirické kritérium pro výzkumné otázky

Z dobře formulovaných otázek musí být jasné, jaká data jsou zapotřebí k jejich zodpovězení. Důležitá kontrola, jak je každá otázka formulována, spočívá v určení dat, která jsou zapotřebí pro její zodpovězení. V kvantitativním šetření kladené otázky operacionalizují proměnné a tvoří s nimi dotazník šetření.[20]

Je třeba navrhnout konceptuální rámec, který definuje vztahy mezi proměnnými a ne pouze izolovaný popis. Je to způsob přemýšlení o proměnných. Získaná data, která jsou

převedena na proměnné, mají mezi sebou určité vztahy. Pomocí těchto vztahů jsou dosaženy odpovědi na otázky průzkumu. Aby byly odpovědi považovány za relevantní a uspokojivé, je třeba tyto vztahy stanovit na základě vhodně podaných otázek.

Hlavním požadavkem na otázky je jejich srozumitelnost a jednoznačnost. Otázky mohou být celkem trojího druhu.

Za prvé to jsou otevřené otázky. Zde nejsou respondentovi nabídnuty žádné možnosti odpovědí. Respondent se může vyjádřit jakkoliv. Za výhodu zde může být považováno neomezování respondenta do nějakých mezí. Naopak pro tazatele to může být nevýhoda, protože respondentovi odpovědi nemusejí přinášet odpovědi a data takového charakteru, který by umožňovaly vytvoření patřičné vazby mezi daty k získání konečných výsledků. U otevřených otázek se může jednak jak o nějakou textovou formu odpovědi, tak třeba o grafické vyjádření.

Za druhé to jsou uzavřené otázky. U tohoto způsobu je respondentovi nabídnuta konkrétní odpověď. Respondent se může rozhodnout mezi jednou autoritativní odpovědí, nebo může být kombinováno více nabízených variant. Výhodou oproti otevřeným otázkám je právě možnost jakéhosi směrování respondenta do oblasti odpovědí, které jsou potřebné k vytvoření patřičných vazeb mezi daty a které mnohem lépe vystihují cíl. Je ovšem třeba volit takové odpovědi, které budou maximálně výstižné a u respondenta nevyvolají pocit, že je nucen volit z odpovědí, které by normálně nevolil.

Jako třetí možnost je kombinace obou předchozích druhů otázek. To znamená, že jsou respondentovi nabídnuty určité možnosti, ze kterých může volit odpověď. Pakliže mu žádná z nabízených odpovědí nevyhovuje, má možnost napsat svou vlastní odpověď.

3.5 Pozorování

V tomto případě se jedná o podobnou situaci jako na obrázku (Obrázek 9), kdy pozorovatel (výzkumník) pouze sleduje chování či vlastnosti sledované jednotky (tazatel).

Je předpokládána objektivita v podobě nezávislosti pozorovatele i objektu tak, že se vzájemně neovlivňují, nepůsobí na sebe. Jsou-li uvedené předpoklady splněny, je považována situace pozorování za normální, objektivní formu získávání informací. Podstatou techniky pozorování je tedy evidence, registrace vlastností a chování sledovaných jednotek za použití vhodného sledovacího zařízení.[21]

Pozorování je pravděpodobně jednou z nejtěžších metod sběru dat v kvalitativním výzkumu. V literatuře najdeme několik variant pozorování, avšak základním typem je zúčastněné pozorování. Zúčastněné pozorování můžeme definovat jako dlouhodobé, systematické a reflexivní sledování probíhajících aktivit přímo ve zkoumaném terénu. Účelem pozorování je deskriptivně zachytit, co se děje a jak vypadá daná situace.[22]

Pozorování přímé je založeno na bezprostředním sledování činnosti. Naopak pozorování nepřímé je charakterizováno pozorováním ze záznamu.

3.6 Experiment

Je provedena změna na sledovaných datech. Následně je pozorován výsledek tohoto experimentu. Evidované reakce považujeme za závislou proměnnou na provedené změně. Problém experimentu spočívá především v tom, že výsledné chování je ovlivněno i jinými změnami v okolním prostředí. Proto je experiment používán zřídka.

- Experiment v terénu – je uskutečněn přímo ve zkoumaném prostředí
- Experiment laboratorní – které probíhají ve zvlášť organizovaném a umělém prostředí.

Nejvíce vypovídající hodnoty dává experiment v terénu. Ale není možné vždycky takový experiment uskutečnit, proto je někdy třeba vytvořit modelové prostředí a experiment pouze simulovat v laboratorních podmínkách. Důvody můžou být jak finanční, tak třeba nemožnost dosáhnout v terénu skutečných podmínek provozu.

II. PRAKTICKÁ ČÁST

4 VYUŽITÍ PROGRAMOVACÍCH JAZYKŮ

V této části diplomové práce je provedena praktická část zadání. Požadavek na využití programovacích jazyků v jednotlivých odvětvích byl rozšířen o některá další data, která přímo souvisejí s používáním programovacích jazyků v praxi.

Samotné používání programovacího jazyka je podmíněno i dalšími aspekty, které musejí být brány v úvahu. V dnešní době je na programátory kladen mnohem větší důraz na komplexnější znalosti oboru.

Vlivem neustále se zrychlujícího pokroku v různých oblastech života vznikají stále nové a nové technologické novinky. Je snaha využívat všech moderních technologií, které můžou usnadňovat práci při samotné tvorbě software. Vznikají další hardwarové platformy, jejichž znalost je nezbytná součást vědomostí programátora.

Nové frameworky usnadňují vývojářům práci tím, že zjednodušují užívání programovacího jazyka a dalo by se říci, že posouvají programování do vyšších úrovní abstrakce.

Není v lidských silách ovládat všechny technologické aspekty, které se můžou používat při vývoji software. Z toho důvodu je taky důležité, aby se potenciální programátor orientoval v týmovém vývoji produktu. Mnohdy jsou třeba bohaté zkušenosti, které přináší patřičnou jistotu a vyšší pravděpodobnosti bezproblémového vývoje ve zvoleném programovacím jazyce.

4.1 Průzkum trhu práce

V případě průzkumu trhu práce byla použita data, která byla zjištěna během měsíců listopad 2010 až únor 2011. Samotnými zdroji dat byly servery zabývající se pracovními nabídkami. Zde byly postupně zaznamenávány požadavky na budoucí zaměstnance a jejich znalosti.

Byly vybrány následující servery:

- http://www.sprace.cz/
- http://www.jobs.cz/
- http://www.prace.cz/

Důležitým aspektem bylo vyřazení nabídek, které pocházely od pracovních agentur. V tomto případě totiž nešlo zjistit údaje o skutečném zadavateli a nebylo možné rozlišit duplicity.

Samozřejmě byl zaznamenán i požadovaný počet pracovních míst na danou pozici v konkrétní firmě.

Jak již bylo zmíněno na začátku kapitoly, průzkum probíhal koncem roku 2010 a začátkem roku 2011. Během této doby bylo nasbíráno 300 záznamů, což představovalo velkou část nabídky pracovních míst programátorů. V nabídkách byly podchyceny opravdu velké firmy jako třeba Seznam.cz přes firmy střední velikosti až po menší vývojářské týmy.

4.1.1 Sbíraná data

Parametry, které byly u tohoto průzkumu zaznamenány, jsou popsány v následujících bodech.

První kritérium byla oblast činnosti. Zde byly zaznamenávány konkrétní odvětví průmyslu a hospodářství, na které byly specializovány dané firmy, případně pro jakou oblast byli budoucí programátoři hledáni. Jednalo se o bankovní služby, pojišťovnictví, ekonomiku, státní správu, telekomunikace, automatické řízení a podobně.

Druhé kritérium bylo působnost společnosti. V tomto případě byly společnosti rozděleny do dvou oblastí. První oblast byly společnosti, které jsou deklarovány jako ryze české. Druhá skupina byly společnosti, prezentující se jako mezinárodní či s velkou mezinárodní působností.

Třetí kritérium byl požadovaný programovací jazyk. Samotným tématem diplomové práce je průzkum užívání programovacích jazyků. V tomto bodě se samozřejmě jednalo o podchycení požadavků na znalosti programovacích jazyků u zájemců o práci programátora v dané firmě. Mnohdy byly požadavky na znalost více programovacích jazyků. Tyto tedy byly seřazeny dle priorit budoucího zaměstnavatele. Programovací jazyk nejvyšší priority potom dostal nejvíce bodů do výsledného hodnocení. Zbývající jazyky potom dostaly odstupňované snižované bodové hodnocení a jazyk s nejmenší prioritou dostal nejmenší počet bodů. Konkrétně jazyk na prvním místě byl hodnocený čtyřmi body, na druhém místě dostal dva body a zbývající místa byla hodnocena jedním bodem. Byl

zaznamenáván i značkovací jazyk HTML, který sice nepatří mezi opravdové programovací jazyky, ale stále je zájem o jeho znalosti.

Čtvrté sledované kritérium bylo požadované vzdělání. Není pravidlem, že požadavky byly podmiňovány vysokoškolským vzděláním v oboru, případně vysokoškolským vzděláním všeobecným. Mnohdy bylo vzdělání označeno jako nepodstatné. V těchto případech byly evidentně preferovány zkušenosti budoucích programátorů.

Páté kritérium byl požadavek na praxi. Zkušenosti, které přináší praxe při vývoji software, jsou velmi důležité. Nebylo výjimkou, kdy byla u potenciálního programátora požadována minimální praxe 5 let. Skutečně existují oblasti tvorby software, kde je kromě znalosti daného programovacího jazyka taky nezbytná praxe na podobných pozicích či projektech. Složitost některých vyvíjených aplikací nedovoluje svěřit tuto oblast nezkušeným programátorům, protože by mohl nastat nečekaný problém v tempu vývoje anebo by výsledná aplikace nemusela pracovat správně. Takové chyby mívají špatné následky na renomé vývojové firmy. Složitá aplikace napsaná nezkušeným programátorem pak může způsobit nepříjemné komplikace i konečnému zákazníku.

Šesté kritérium byl požadavek na schopnost práce v týmu. U vývoje aplikací je v dnešní době moderní trend společného vývojového týmu a použití speciálních aplikací k tomu určených. Existuje ale i mnoho oblastí, kde práce v týmu není možná z mnoha důvodů. Může to být způsobeno nemožností spolupráce více vývojářů u speciálních hardwarových platforem. V každém případě schopnost pracovat v týmu, komunikovat se členy týmu či schopnost používat verzovací nástroje by dnes měla patřit do výbavy programátorů.

Sedmé kritérium byla cílová platforma. Kde bylo uvedeno, byla zaznamenána i cílová platforma na kterou bude výsledný produkt v daném programovacím jazyce směrován. Znalosti cílové platformy považuji za jednu z nejdůležitějších znalostí programátora, kromě znalosti programovacího jazyka samotného. Bez porozumění architektury, pro kterou je nový systém vyvíjen není možné udělat dobrý software. Je třeba znát funkce zásobníků, správy procesů, důležitá je znalost práce s pamětí na cílové platformě a podobně. V případě speciálních platforem jako jsou třeba programovatelné automaty, je znalost cílového hardwaru pro programátora prakticky nezbytná.

Poslední kritérium bylo doplnění o požadavek dalších znalostí. Zde byly zaznamenány případné požadavky na znalosti technologií, které s vývojem softwaru souvisejí. Jednalo se například o aplikační servery či frameworky.

4.2 Dotazníkový průzkum

Dotazník je jedním z nejběžnějších nástrojů pro sběr dat pro různé typy průzkumů. Skládá se ze série otázek, jejichž cílem je získat názory a fakta od respondentů.[23]

U tohoto typu empirického výzkumu byly postupně obeslány speciálním dotazníkem konkrétní softwarové vývojové firmy, které byly opět vybrány v různých oblastech působnosti.

Dotazník by měl na první pohled upoutat pozornost, nesmí respondenta hned na začátku odradit. Je potřeba se zaměřit především na [24]:

- Srozumitelnost
- Přehlednost a snadnou orientaci
- Jednoduchost vyplňování
- Jazykovou korektnost
- Typografickou úpravu
- Grafickou úpravu

Veškeré otázky byly směřovány tak, aby byly vytvořeny potřebné vzájemné vazby mezi proměnnými. Kromě samotného programovacího jazyka byly zjišťovány podobně jako v kapitole (Kapitola 4.1.1) i údaje a data bezprostředně související se znalostmi programovacích jazyků.

Na začátku dotazníku je vhodné řadit zajímavé otázky, které upoutají pozornost respondenta. Uprostřed jsou otázky stěžejní a na konci dotazníku jsou otázky méně závažné.[24]

4.2.1 Popis zvolených otázek

Zde jsou popsány jednotlivé otázky dotazníkového průzkumu. Jedním z dalších kritérii kromě srozumitelnosti byla taky jednoduchost a rychlost vyplnění. Není možné

předpokládat, že respondent je ochotný věnovat vyplnění dotazníku více času. Pro sběr takto vyplněných dat byl zvolen server www.vyplnto.cz, který se právě na poskytování technologické platformy pro dotazníky specializuje a obsahuje odpovídající grafické a technické zázemí.

První otázka:

Na jakou platformu je cílený váš software?

U této otázky je zjišťován cílový hardware a software pro vyvíjenou aplikaci. Znalost funkce jednotlivých komponent, na kterých bude software spuštěn, je jednou z nejdůležitějších znalostí vývojářů.

U této otázky je snaha zjistit tyto priority. Při porovnání s jinými otázkami tohoto dotazníku lze potom vysledovat vhodnost studia přidružených předmětů u oborů zabývajících se informačními technologiemi na vysokých školách.

V odpovědích byly nabídnuty následující varianty:

- Linux
- Windows
- Mac
- Programovatelné automaty (PLC)
- Servery a síťový hardware
- Mikropočítače a jednočipy
- Embedded systémy
- Mobilní zařízení
- Konzoly
- Jiné platformy

Druhá otázka:

Pro jaké odvětví vyrábíte software?

Empirický výzkum v této diplomové práci je zaměřen na využití programovacích jazyků v rámci jednotlivých hospodářských odvětví. Byla snaha pokrýt co nejširší spektrum

V odpovědích bylo nabídnuto celkem 22 konkrétních odvětví, případně možnost jiného odvětví.

Jednalo se o tato odvětví (viz Tabulka 1):

Ekonomika a obchod	Potravinářství
Bankovnictví a pojišťovnictví	Telekomunikace
Strojírenství	Zdravotnictví
Elektrotechnický průmysl	Státní správa
Sport	Software na zakázku
Školství	Herní software
Stavebnictví	Energetika
Automatizace a řízení	Software pro specializovaný hardware
Automobilní průmysl	Letecký průmysl
Doprava	GIS – geografické systémy
Ochrana objektů, bezpečnost	Oblast IT (informační technologie)

Tabulka 1 – Vybraná odvětví hospodářství v průzkumu

Třetí otázka:

Jaké používáte programovací jazyky k tvorbě software?

Hlavním poznatkem z dotazníku bylo získání informací o konkrétním programovacím jazyku a jeho využití v dotazovaných společnostech.

Jelikož je programovacích jazyků nevyčíslitelné množství, byla provedena selekce na nejpoužívanější programovací jazyky s možností volby jiného jazyka, který nebyl uveden v nabízených odpovědích.

Jako navrhované programovací jazyky byly zvoleny následující (viz Tabulka 2):

ASP	LISP
ASP.NET	Objective-C
Assembler (JSA)	Pascal
C	Perl
C#	PHP
C++	Python
Cobol	Ruby
Delphi	VHDL
Fortran	Visual Basic
Java	Visual Basic.NET
JavaScript	

Tabulka 2 – Programovací jazyky empirického průzkumu

Čtvrtá otázka:

Při zahájení vývoje nového produktu upřednostňujete některý z výše uvedených programovacích jazyků?

U čtvrté otázky byly zjišťovány preference jednotlivých jazyků u vývojářů. Právě preference pro konkrétní jazyk jsou jistým směrníkem, kam se může ubírat zájem o programovací jazyky do budoucnosti.

Volba konkrétního programovacího jazyka u aplikace je většinou rozhodnutí, které nelze jednoduše později změnit. Samotný vývoj aplikace a vydávání novějších verzí je záležitost několika příštích let a zvolený programovací jazyk je tedy volen s dlouhodobým předpokladem využívání.

Odpovědi byly z těchto možností:

- Jeden jazyk (který)
- Neupřednostňuji žádný jazyk

Pátá otázka:

Z jakého důvodu upřednostňujete výše uvedený programovací jazyk?

V případě kladné odpovědi na čtvrtou otázku byl zobrazen dotaz na udání důvodu upřednostňování zvoleného programovacího jazyka.

Těchto důvodů může být velké množství, proto byly redukovány do větších skupin, ze kterých je jasný hlavní motiv pro případné upřednostnění konkrétního programovacího jazyka.

Tyto skupiny lze charakterizovat následovně:

- Rychlost programu
- Rychlejší vývoj aplikace v porovnání s jinými jazyky
- Nenáročný na prostředky hardware (paměť, procesor)
- Spolehlivost a robustnost
- Není třeba řešit složitosti při návrhu aplikace jako v jiných jazycích
- Výhody automatické správy přidělených prostředků
- Užitečné knihovny
- Jsou k dispozici kvalitní frameworky
- Programovací jazyk je jednoduchý na používání a zaučení
- V jiném programovacím jazyku nelze naprogramovat některé specifické úlohy
- Máme v tomto programovacím jazyce velké zkušenosti
- Daný programovací jazyk má dobrou budoucnost
- Je ovládaný většinou programátorů
- Náš software je v tomto jazyce vyvíjen a nelze přejít na jiný
- Finance
- Případný jiný důvod

Šestá otázka:

Jaký programovací jazyk byste doporučili k důkladné výuce na VŠ?

Výuka programovacích jazyků je záležitost mnoha let. Není ale možné naučit se detailněji více jazyků. I když programátoři mají všeobecné znalosti z více programovacích jazyků, zaměřit se a detailněji poznat nelze všechny ve stejné míře. U některých jazyků jako je například C++ je v mnoha pramenech uváděna doba až 10 let na hluboké proniknutí do zákonitostí a dovedností zvoleného programovacího jazyka.

Jako dobrý základ pro budoucí směřování programátora je proto vhodné zaměřit se při výuce programovacích jazyků na vysokých školách na takové typy, které jsou v praxi vyžadovány.

Jako odpověď je zde požadován konkrétní programovací jazyk.

Sedmá otázka:

Chystáte se v nejbližších měsících změnit priority u používaných programovacích jazyků?

Tato otázka úzce souvisí s předchozími třemi otázkami.

Odpovědi, které byly nabídnuty, jsou:

- Ano
- Nevím
- Ne.

Osmá otázka:

Používáte při vývoji software verzovací systém?

Trendem posledních let se stalo používání verzovacích systémů. Jedná se o produkty, které usnadňují vývoj v týmu. Díky nasazení verzovacích systémů můžou vývojáři pracovat na vývoji software společně a případně nějaké chyby se lze vracet k předchozím verzím zdrojového kódu.

Tato společná práce v týmu ovšem přináší další požadavky na znalosti programátora. Je třeba umět vyřešit konflikty při sestavování hlavní vývojové větve. Je třeba dodržovat dané pravidla syntaxe při psaní kódu a podobně.

Využívání verzovacích systémů by mohlo být zavedeno i na vysokých školách při zadávání ročníkových či semestrálních prací. Student by měl možnost seznámit se s tímto způsobem vývoje aplikace.

Jako odpověď u této otázky jsou nabízeny možnosti:

- Ano
- Ne
- Chystáme se.

Devátá otázka:

Požadujete po uchazečích o práci vysokoškolské vzdělání technického směru?

Požadavek vysokoškolského vzdělání je v dnešní době důležitý. Právě kvalita výuky programovacích jazyků na vysokých školách může u této otázky hrát významnou roli.

Bylo možné volit z následujících odpovědí:

- Ano
- Ne
- Vzdělání není důležité, ale je výhodou

Desátá otázka:

Je dosažená praxe v oboru u uchazeče důležitější, než vzdělání na škole zaměřené na IT?

Dosažená praxe je velmi důležitá. V oblastech nízkoúrovňového programování je prakticky nedílnou součástí výbavy programátora.

Volit bylo možné z následujících možností:

- Ano
- Spíše ano
- Nevím
- Spíše ne
- Ne

Jedenáctá otázka:

Jaká je v současnosti úroveň znalostí vámi požadovaných programovacích jazyků u uchazečů o práci u vás?

Otázka je směřována na kvalitu a znalosti nových uchazečů na post vývojářů software.

Odpovědi bylo možné zvolit z následujících možností jako ve škole:

- Výborný
- Chvalitebný
- Dobrý
- Dostatečný
- Nedostatečný

Dvanáctá otázka:

Jak dlouho se věnujete vývoji softwaru?

Samotné stáří vývojové firmy je ukazatel, jaké zkušenosti již má na trhu. Může to být ukazatel na kvalitu odpovědí. Zvolil jsem hranici pěti let.

Možné odpovědi tedy byly:

- Méně než 5 let
- Více než 5 let

Třináctá otázka:

Je podle vašeho názoru v současnosti dostatek kvalitních programátorů?

To je otázka, která dává odpověď na mnoho témat. Dostatek kvalitních programátorů může pro daná odvětví a programovací jazyky určovat směřování do dalších let. Při dostatku kvalitních odborníků nenastávají problémy se zpožděním vývoje aplikace tak často, jako u úzkoprofilových programátorských odvětví.

Odpovědi tedy byly vybírány z následujících možností:

- Ano
- Spíše ano
- Nevím
- Spíše ne
- Ne

Čtrnáctá otázka:

Obrat firmy?

U tohoto typu otázky byl cíl směřovaný k zjištění, jestli je používaný programovací jazyk závislý na velikosti obratu ve firmě. Vývoje aplikací u některých typů programovacích jazyků můžou být nákladné. Z toho vyplývá, že výsledná cena produktu musí odpovídat vynaloženým nákladům.

Taky se dá říct, že se stoupajícím obratem firmy si lze dovolit rozšířit využívání technologických novinek v oblasti, pro kterou je směřovaný výsledný produkt.

Odpovědi byly rozděleny na 3 varianty. Obrat do 10 milionů Kč byla první možnost, druhá možnost byl obrat nad 10 milionů Kč a poslední možnost byla firma bez obratu. U poslední možnosti se mohlo jednat třeba o státní podnik.

- Menší než 10 milionů Kč
- Větší než 10 milionů Kč
- Bez obratu
- Nevím

Patnáctá otázka:

Zázemí společnosti?

Zde se jednalo o zjištění, zda se firma opírá o zázemí nadnárodní společnosti nebo si svou pozici na trhu musela vybudovat samostatně jako ryze česká společnost.

Jakými prostředky dosáhli současného stavu při jejich vlastnických vztazích lze vysledovat z dalších otázek.

K výše uvedeným možnostem byla ještě přidána možnost odpovědi, kdy je vlastníkem stát.

- Česká společnost
- Nadnárodní společnost
- Státní podnik

Šestnáctá otázka:

Počet zaměstnanců?

Určení množství zaměstnanců společnosti může vypovídat o schopnostech firmy při potenciálním zahájení vývoje nového produktu. S větším počtem zaměstnanců lze zajistit větší komfort pro zákazníky, ale je tady potřeba kvalitních a dobře prodávaných softwarových komponent.

Spolu související je zde i otázka rychlosti vývoje. Je možné používat programovací jazyky vyšších úrovní abstrakce, kde jsou urychleny některé procesy a tím i vývojový cyklus aplikace.

Byly nabídnuty dvě možnosti. První byla společnost do padesáti zaměstnanců a druhá možnost byla společnost s více než padesáti zaměstnanci, což je považováno za jakousi hranici, kdy už může být hovořeno o větší společnosti.

- Více než 50 zaměstnanců
- Méně než 50 zaměstnanců
- Nevím

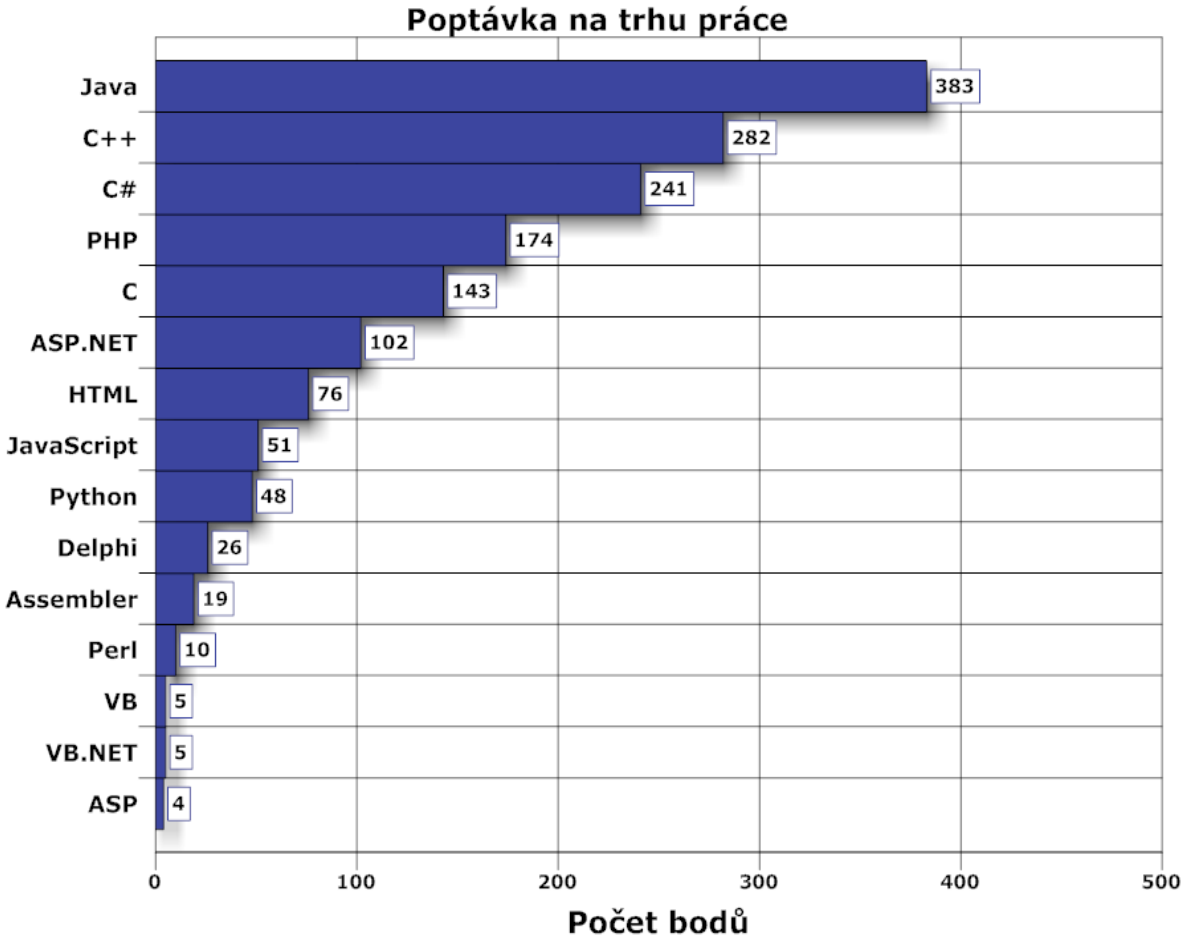
4.3 Výsledky průzkumu trhu práce

V následujících grafech jsou shrnuty získané výsledky, které se opírají o vzájemné vazby mezi jednotlivými proměnnými. Zjištěné výsledky aktuálně odráží současné požadavky. Některé firmy požadovaly více zaměstnanců na danou pozici. Někdy se taky jednalo o zaměstnání na různých projektech u stejného zaměstnavatele, ale s odlišnými požadavky na znalost programovacích jazyků.

Všechny tyto aspekty byly zohledněny při sběru dat.

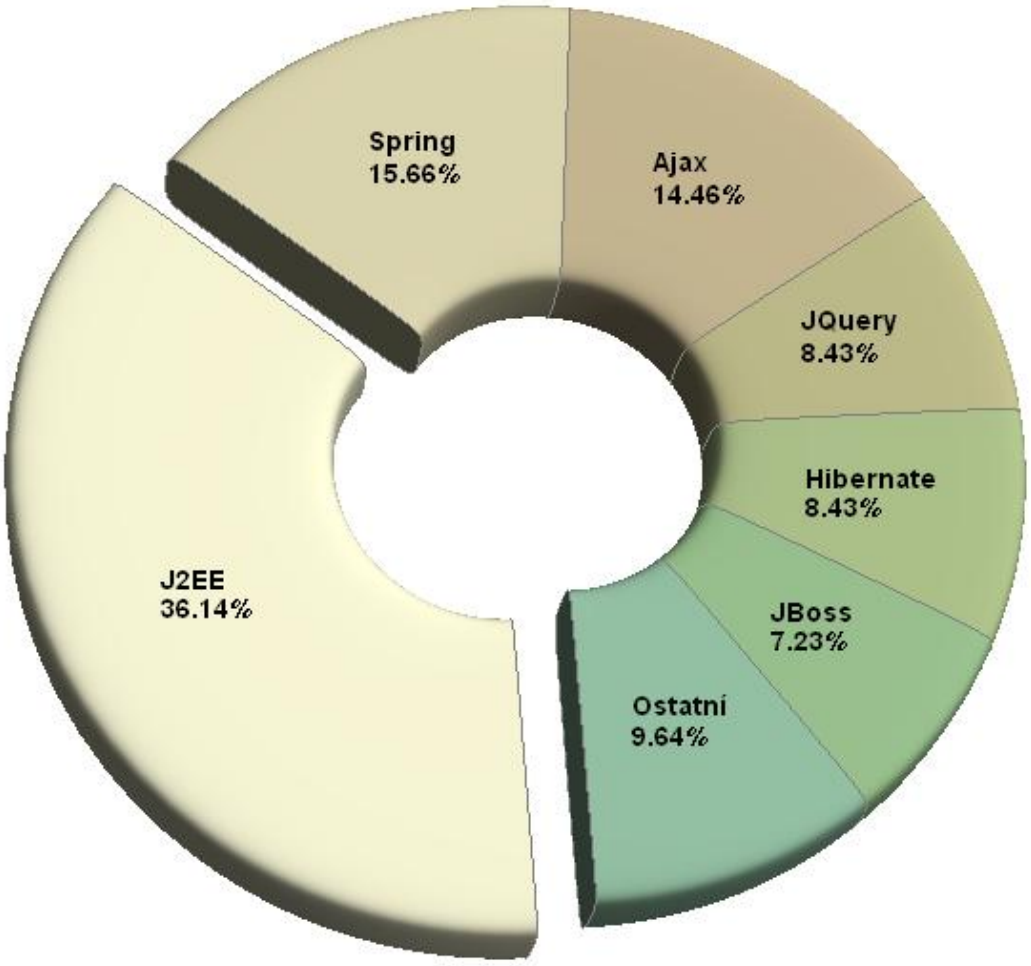
4.3.1 Požadavky na znalost programovacích jazyků

V následujícím obrázku (Obrázek 11) je znázorněno celkové pořadí preference jednotlivých programovacích jazyků na trhu práce bez dalších omezujících okolností.



Obrázek 11 – Požadavky trhu práce na znalost programovacích jazyků

Z obrázku (Obrázek 11) vyplývá, že v současnosti je na trhu práce největší zájem o objektově orientované jazyky. Jako nejúspěšnější se jeví jazyk Java, který je sice interpretovaný a tedy pomalejší než neinterpretované jazyky, ale přesto je nejžádanějším jazykem na znalosti programátorů. Je to dáno mimo jiné i tím, že se v posledních letech velmi zapracovalo na zrychlení interpretovaných jazyků a také zaváděním aplikačních serverů, které vnášejí další vrstvu abstrakce mezi interpretované jazyky a rozhraní operačního systému.



Obrázek 12 – Požadavky na znalosti souvisejících technologií

Na obrázku (Obrázek 12) je ukázán výsledek průzkumu zaměřeného na znalosti souvisejících technologií. Jak již bylo zmíněno v kapitole (Kapitola 4.1.1), znalost programovacích jazyků mnohdy souvisí i se znalostmi aplikačních technologií. Zde jsou zaznamenány tyto požadavky v případě, že byly zadavatelem prezentovány. Požadavky na znalosti technologií se vyskytovaly v 1/3 publikovaných požadavků na schopnosti programátorů. Je vynechána technologie .NET (nebo jeho multiplatformní open source varianta Mono), která je požadována u všech programovacích jazyků založených na tomto frameworku, to znamená ASP.NET, C# a VB.NET.

4.3.2 Průzkum trhu v jednotlivých sektorech

V jednotlivých oblastech hospodářství, která byla uváděna při nabídkách práce programátorů, byly zjištěny následující priority znalostí programovacích jazyků.

Automatizace, řízení, elektrotechnika	
C	25,22 %
Java	24,35 %
C++	19,13 %
JSA (Assembler)	14,78 %
C#	9,56 %
Delphi	5,22 %
Perl	0,87 %
Python	0,87 %

Tabulka 3 – Průzkum trhu v oblasti automatizace

V oblasti automatizace, řízení a elektrotechniky (viz Tabulka 3) převažují jazyky nižší úrovně, kde jedinou výjimkou je jazyk Java, který se začal využívat i v těchto oblastech programování, které byly dříve výhradní doménou nízkoúrovňových jazyků.

Doprava a letectví	
C++	31,58 %
Java	29,82 %
C	28,07 %
C#	10,53 %

Tabulka 4 – Průzkum trhu v oblasti dopravy

V oblasti dopravy a letectví (viz Tabulka 4) jsou nejvíce uplatňovány typicky jazyky nižších úrovní, jako je jazyk C++ a C, dále je často užíváno programovacího jazyka Java. Letecký průmysl není v současnosti v naší zemi příliš rozšířený, a tak výroba software pro toto odvětví je v porovnání s jinými menší.

Automobilní průmysl	
Java	40 %
C#	30 %
C++	20 %
VB.NET	5 %
ASP.NET	5 %

Tabulka 5 – Průzkum trhu v oblasti automobilního průmyslu

U tvorby software pro automobilní průmysl (viz Tabulka 5) jsou požadovány především znalosti z oblasti interpretovaných jazyků a platformy .NET.

Energetika	
JSA (Assembler)	25,53 %
C	23,41 %
C#	17,02 %
C++	14,89 %
Java	10,64 %
ASP.NET	8,51 %

Tabulka 6 – Průzkum trhu v oblasti energetiky

Oblast energetiky (viz Tabulka 6) je velmi náročná na spolehlivost a hlavně rychlost software. Proto jsou zde nejvíce uplatňovány znalosti jazyků nejnižší úrovně. Dostatečně uplatňován je zde i jazyk C#.

U energetiky se jedná hlavně o zařízení, která řídí složité systémy dodávek elektřiny, můžou hlídat systémy v elektrárnách a podobně.

CAD, CAM, Strojírenství	
C#	32,14 %
ASP.NET	17,85 %
C	14,29 %
C++	14,29 %
Java	14,29 %
VB.NET	7,14 %

Tabulka 7 – Průzkum trhu v oblasti strojírenství

V oblasti strojírenství, návrhových a mapových podkladů (viz Tabulka 7) jsou na trhu práce největší požadavky na znalosti platformy .NET, konkrétně je využíván programovací jazyk C# a ASP.NET.

Zdravotnictví	
C++	57,15 %
Java	19,05 %
C#	9,52 %
C	4,76 %
VB.NET	4,76 %
ASP.NET	4,76 %

Tabulka 8 – Průzkum trhu v oblasti zdravotnictví

Oblast zdravotnictví (viz Tabulka 8) vyžaduje především programátory se znalostí jazyka C++. Jazyk C++ je považován za velmi spolehlivý a rychlý. Ve zdravotnictví jsou tyto vlastnosti u některých druhů software přímo nezbytné.

Telekomunikace	
C++	31,11 %
Java	24,44 %
C#	16,67 %
C	12,22 %
PHP	6,66 %
HTML	3,33 %
JavaScript	1,67
ASP.NET	1,67
Perl	1,67
Python	0,56

Tabulka 9 – Průzkum trhu v oblasti telekomunikací

V oblasti tvorby software pro telekomunikační průmysl (viz Tabulka 9) jsou v naprosté většině uplatňovány objektově orientované jazyky.

Školství, sport, státní správa	
Java	39,29 %
C++	28,57 %
ASP.NET	21,43
C#	7,14
JavaScript	3,57

Tabulka 10 – Průzkum trhu v oblasti školství

Tvorba software pro oblast školství (viz Tabulka 10) využívá nejvíce programovací jazyk Java a C++. Možnosti jazyka Java se za poslední léta hodně zlepšily. Znalosti tohoto jazyka jsou v oblasti tvorby software pro školství sport a státní správu důležité.

Tvorba software na zakázku	
Java	46,32 %
C#	17,37 %
ASP.NET	12,63 %
C	6,32 %
PHP	5,26 %
C++	3,68 %
Delphi	2,63 %
JavaScript	2,11 %
Python	2,11 %
HTML	1,57 %

Tabulka 11 – Průzkum trhu v oblasti tvorby SW na zakázku

Oblast tvorby software na zakázku (viz Tabulka 11) je specifická tím, že je zde vytvářen software pro široké spektrum uživatelů a odvětví. Mnohdy záleží na rychlosti, jakou je firma schopna software dodat či vyvinout. Z toho důvodu je nejpoužívanější vysokoúrovňový jazyk Java, který ulehčuje a zrychluje práci na vývoji software tím, že již obsahuje potřebné automatické nástroje mající na starosti správu nevyužívaných zdrojů a podobně. Dalším hodně používaným jazykem je C#, opět využívající podpory frameworku .NET.

V oblasti tvorby software na zakázku se ale můžou vyskytovat specifická zadání, která si vynucují užití konkrétního programovacího jazyka. Záleží totiž na tom, jakou úlohu má budoucí aplikace plnit. V případě, že se jedná o ovládání výrobního stroje pomocí programovatelného automatu, je předpoklad, že bude použit programovací jazyk C, který je pro tyto účely nejvhodnější.

Software pro informační technologie	
Java	18,29 %
PHP	17,57 %
C++	17,29 %
C#	14,43 %
C	8,43 %
HTML	6,57 %
ASP.NET	6,14 %
Python	4,14 %
JavaScript	3,43 %
Delphi	1,14 %
Perl	0,86 %
Visual Basic	0,71 %
ASP	0,57 %
JSA (Assembler)	0,29 %
VB.NET	0,14 %

Tabulka 12 – Průzkum trhu v oblasti tvorby software pro IT

Zájem o znalosti programovacích jazyků při tvorbě software zaměřeného na využití přímo v oblasti informačních technologií (viz Tabulka 12) je široký. Z výsledků vyplývá, že žádný jazyk vysloveně nepřevyšuje ostatní a na prvních místech jsou tyto hodnoty přibližně vyrovnané. Opět se uplatňuje objektově orientované programování.

Herní software	
C++	25,81 %
HTML	20,95 %
JavaScript	19,38 %
PHP	12,90 %
C#	6,45 %
C	6,45 %
Java	6,45 %
ASP.NET	1,61 %

Tabulka 13 – Průzkum trhu v oblasti tvorby herního software

Oblast herního software (viz Tabulka 13) je náročná na rychlost. Z výzkumu vyplynulo, že na klasický desktopový či konzolový herní software je nejvíce vyžadována znalost jazyka C++, který je pro tyto účely vhodný a je neinterpretovaný.

Naopak pro oblast herního software určeného pro provoz na webových technologiích jsou uplatňovány klasické programovací a značkovací jazyky jako jsou JavaScript, PHP a HTML.

Je třeba zdůraznit, že u herního softwaru, který je určený pro desktopová nebo konzolová prostředí je třeba znát i mnohé další technologie, které vytváří součinnost s daným programovacím jazykem. Nejčastěji to bývají grafické, zvukové a jiné knihovny, které pomáhají akcelerovat běžící aplikaci. Jako příklad lze jmenovat například DirectX nebo OpenGL a jiné.

Bankovnictví a pojišťovnictví	
Java	57,69 %
C#	21,54 %
ASP.NET	4,62 %
HTML	4,62 %
C++	3,85 %
PHP	3,08 %
JavaScript	2,30 %
Delphi	1,53 %
C	0,77 %

Tabulka 14 – Průzkum trhu v oblasti bankovnictví a pojišťovnictví

V oblasti nakládání s finančními prostředky jako je bankovnictví a pojišťovnictví (viz Tabulka 14) je vyžadována znalost hlavně interpretovaných programovacích jazyků. Nejvíce je využíván jazyk Java a dále jsou to jazyky na platformě .NET.

U tohoto typu software je třeba důsledně dbát na robustnost a odolnost. Při použití interpretovaných jazyků se těchto vlastností dá docílit. Je to zapříčiněno tím, že tyto jazyky používají při své činnosti jakousi mezivrstvu, která obsahuje patřičné prostředky pro přístup k funkcím cílové platformy. Předpokládá se, že tato mezivrstva, která je dodána poskytovatelem daného interpretovaného jazyka, je maximálně vyladěná. Díky tomu se můžou vývojáři soustředit na nové funkce u svého softwaru a nemusí řešit otázky nízkoúrovňových přístupů a potenciálních bezpečnostních rizik.

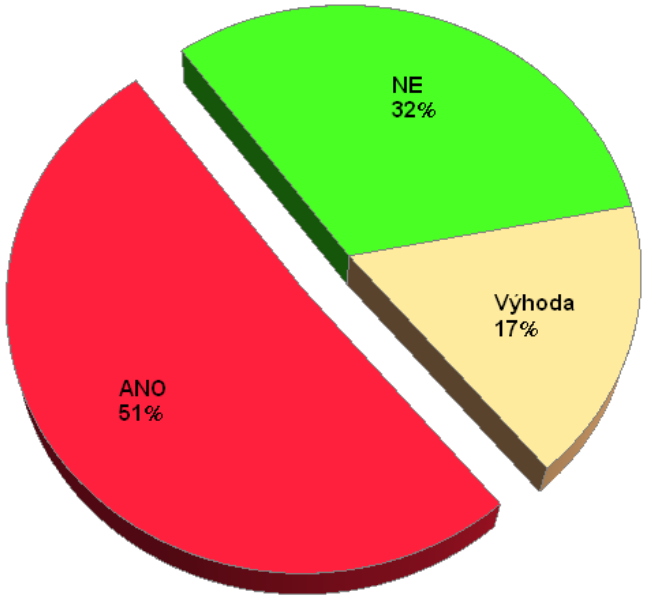
Ekonomika	
PHP	21,86 %
Java	20,47 %
C#	16,74 %
C++	9,30 %
Delphi	7,44 %
ASP.NET	6,05 %
HTML	5,58 %
Python	5,12 %
JavaScript	3,25 %
Visual Basic	1,86 %
VB.NET	1,40 %
Perl	0,93 %

Tabulka 15 – Průzkum trhu v oblasti tvorby SW pro ekonomické účely

Software v oblasti ekonomiky (viz Tabulka 15) je v dnešní době také ve velké míře provozován jako serverové řešení. Z tohoto důvodu je mezi nejžádanějšími jazyky PHP a jazyk Java, které jsou velmi vhodné pro nasazení v rámci počítačové sítě a webových technologií.

4.3.3 Požadavky na praxi a vzdělání, schopnosti práce v týmu

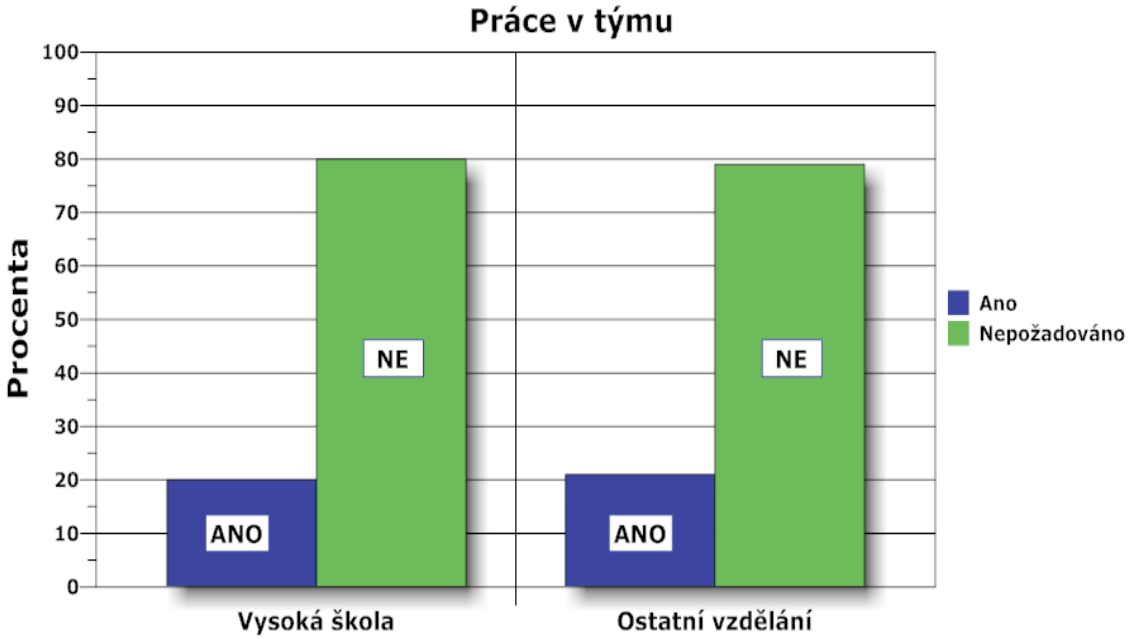
Z celkového počtu 300 sledovaných požadavků na práci programátora byla vysoká škola striktně vyžadována u 94 požadavků, což je přibližně 31 %.



Obrázek 13 – Požadavek na praxi u VŠ vzdělání

V grafu (viz Obrázek 13) je znázorněn výsledek průzkumu v souvislosti s podmínkou dosažení vysokoškolského vzdělání a současně požadavkem na praxi u budoucího uchazeče.

Výsledek průzkumu říká, že praxe v programování je vyžadována u více než poloviny vysokoškolsky vzdělaných programátorů na trhu práce. Délka požadované praxe, pokud byla uvedena, se pohybovala od jednoho roku až do pěti let.



Obrázek 14 – Výsledky průzkumu požadavků práce v týmu

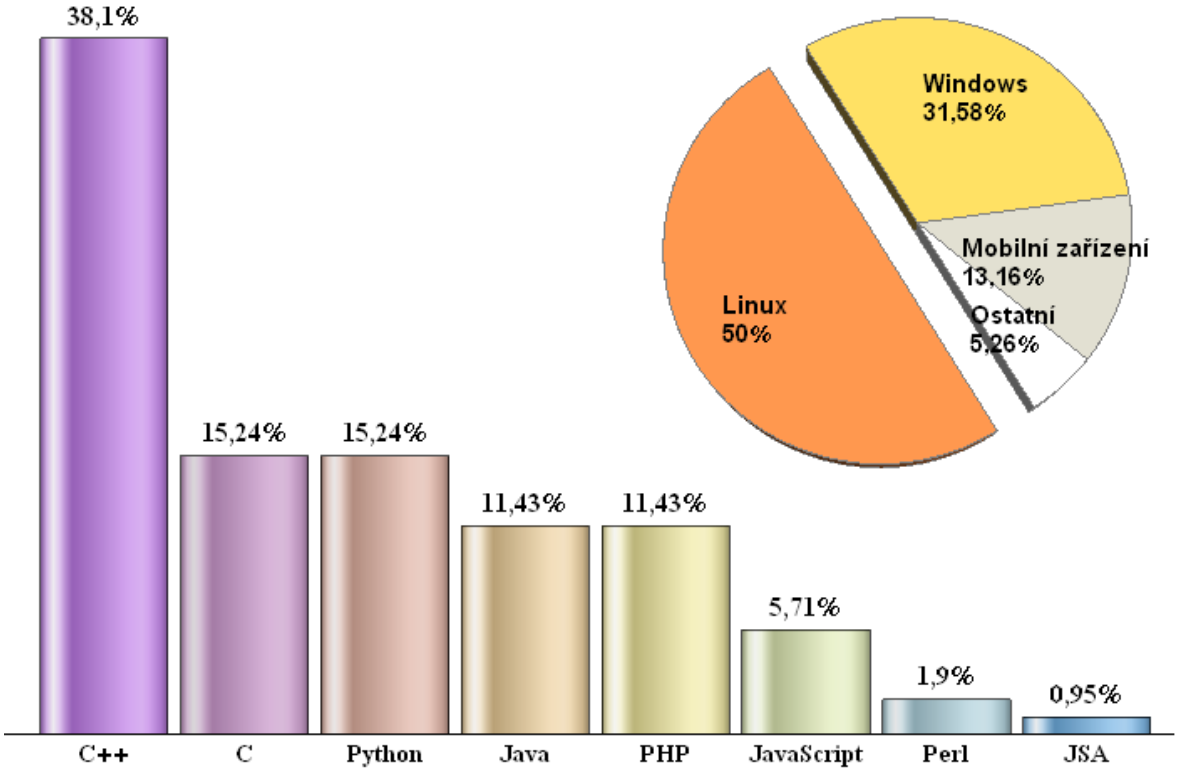
Jedna z důležitých schopností programátora je práce v týmu (viz Obrázek 14). Díky trendu ve využívání verzovacího software jsou tyto schopnosti vyžadovány od nových programátorů. V průzkumu pracovního trhu byly požadavky na práci v týmu podle následujícího grafu. Práce v týmu je vysloveně požadována v menším procentu případů. A nezáleží přitom na dosaženém vzdělání.

4.3.4 Působnost firem na trhu práce z hlediska vlastnictví

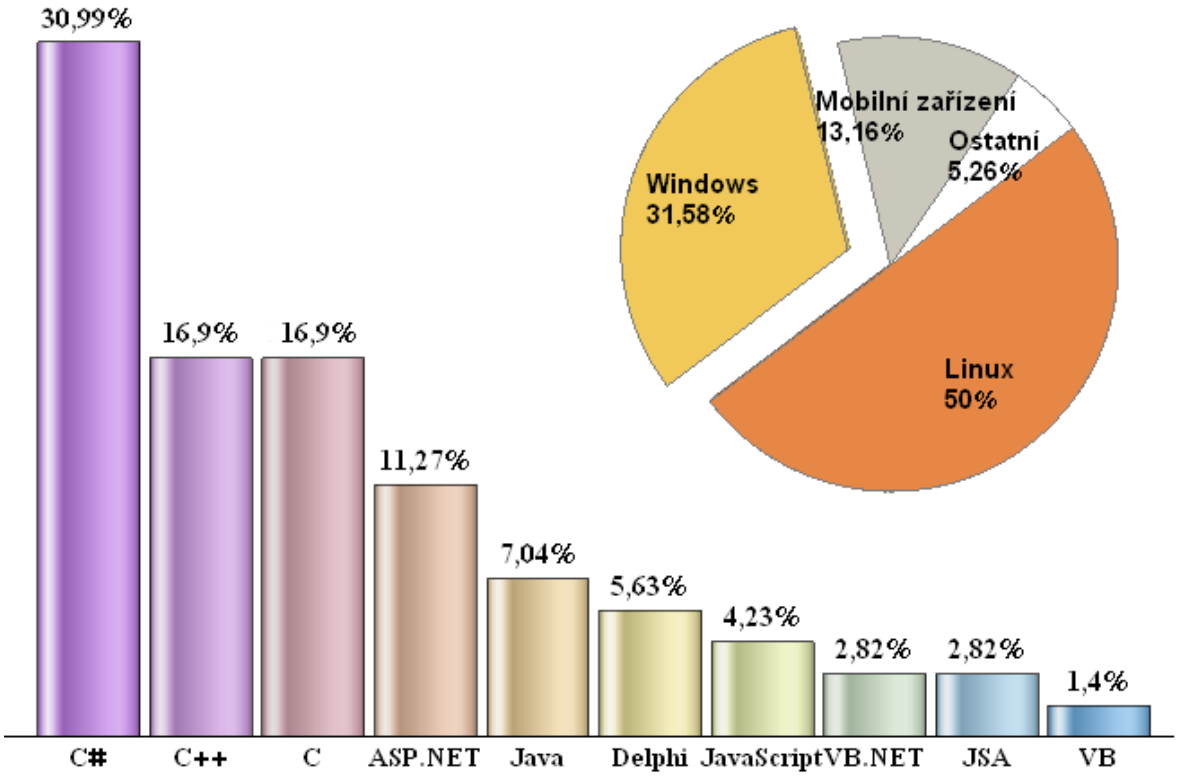
Výsledky průzkumu trhu práce týkají se šíře působnosti firmy. Z celkového počtu 300 firem, které měly požadavky na nové programátory, bylo 120 firem v nadnárodních společnostech. To znamená, že v téměř 59% se jednalo o práci pro ryze české společnosti.

4.3.5 Požadavky na znalosti programovacích jazyků u ryze českých firem podle cílové platformy

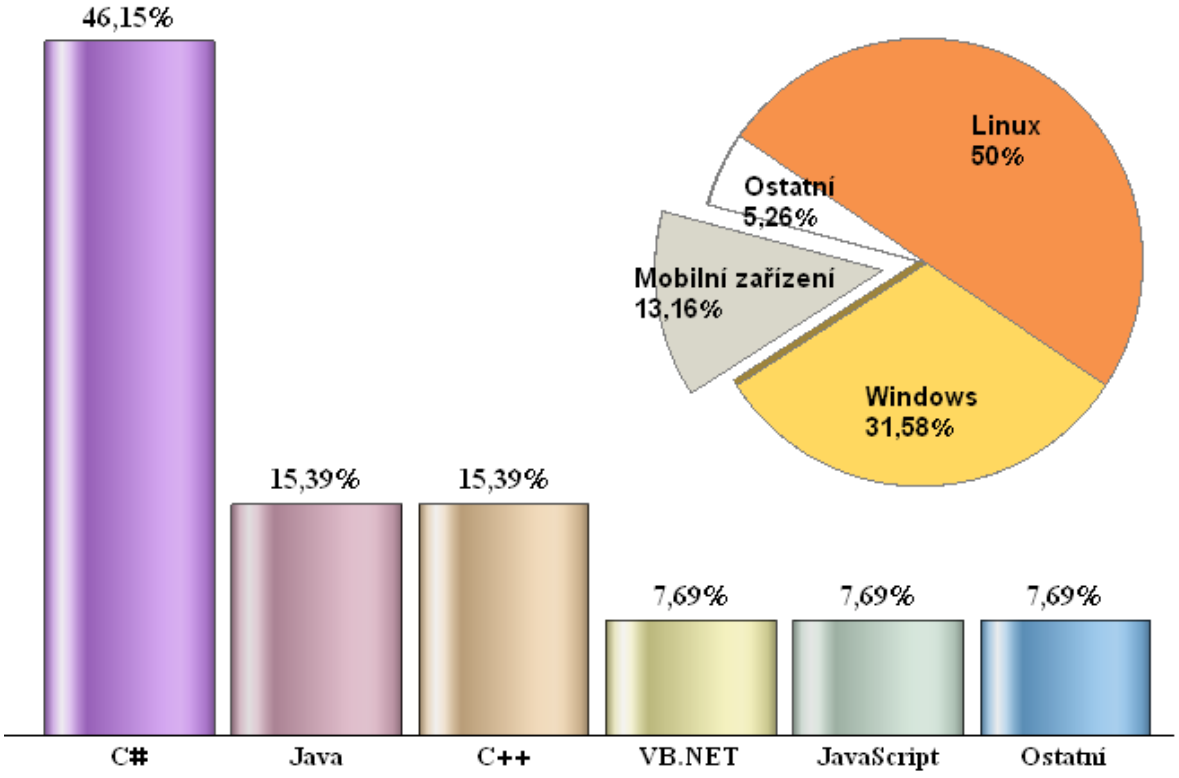
V následujících obrázcích jsou vyznačeny požadavky na znalosti programovacích jazyků u jednotlivých platforem u českých firem. V kruhovém grafu jsou potom vyznačeny celkové podíly jednotlivých platforem.



Obrázek 15 – Požadavky českých společností zaměřených na Linux



Obrázek 16 – Požadavky českých společností zaměřených na Windows



Obrázek 17 – Požadavky českých společností zaměřených na mobilní platformu

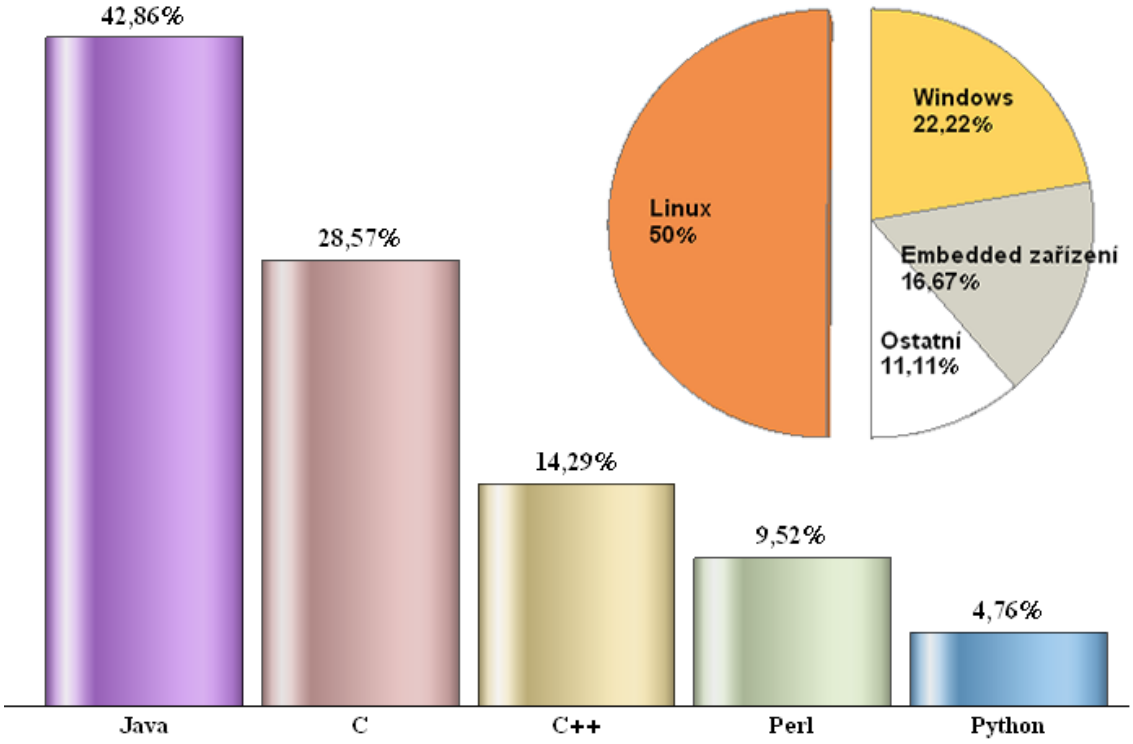
Na obrázku (viz Obrázek 15) je znázorněn zjištěný výsledek požadavku znalostí programovacích jazyků u českých společností, které vyznačily jako své cílové platformy operační systémy na bázi UNIX/Linux. Je zřetelné, že zde převažují programovací jazyky C++ a C. Samotný Linux má k těmto jazykům velmi blízko, dokonce jádro tohoto operačního systému je programováno v jazyce C.

U českých společností, jejichž software je cílený na operační systém z dílny Microsoftu převažuje požadavek na znalost jazyka C# (viz Obrázek 16). Tento jazyk je Microsoftem podporován a vyvíjen, takže tato kombinace se jeví jako ideální.

I v oblasti softwaru cíleného na mobilní platformu (viz Obrázek 17) je velmi oblíbený u českých společností jazyk C#. Samozřejmostí je taky jazyk Java, který je navíc multiplatformní.

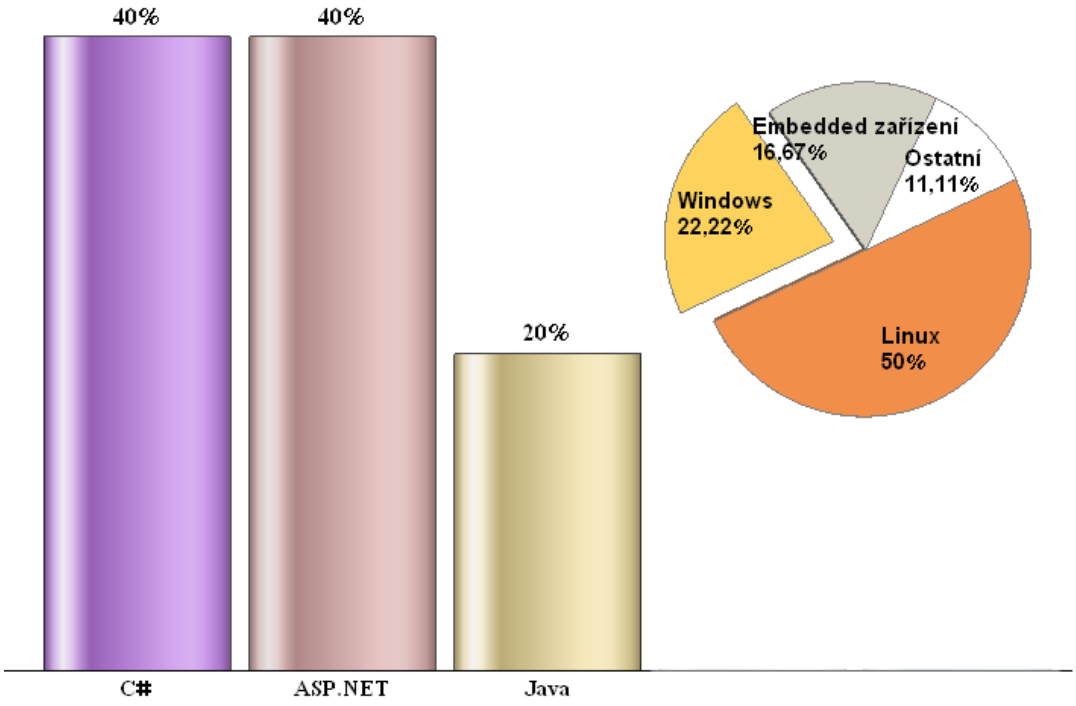
4.3.6 Požadavky na znalosti programovacích jazyků u nadnárodních firem podle cílové platformy

V následujících obrázcích jsou vyznačeny požadavky na znalosti programovacích jazyků u jednotlivých platforem u nadnárodních firem. V kruhovém grafu jsou potom vyznačeny celkové podíly jednotlivých platforem.

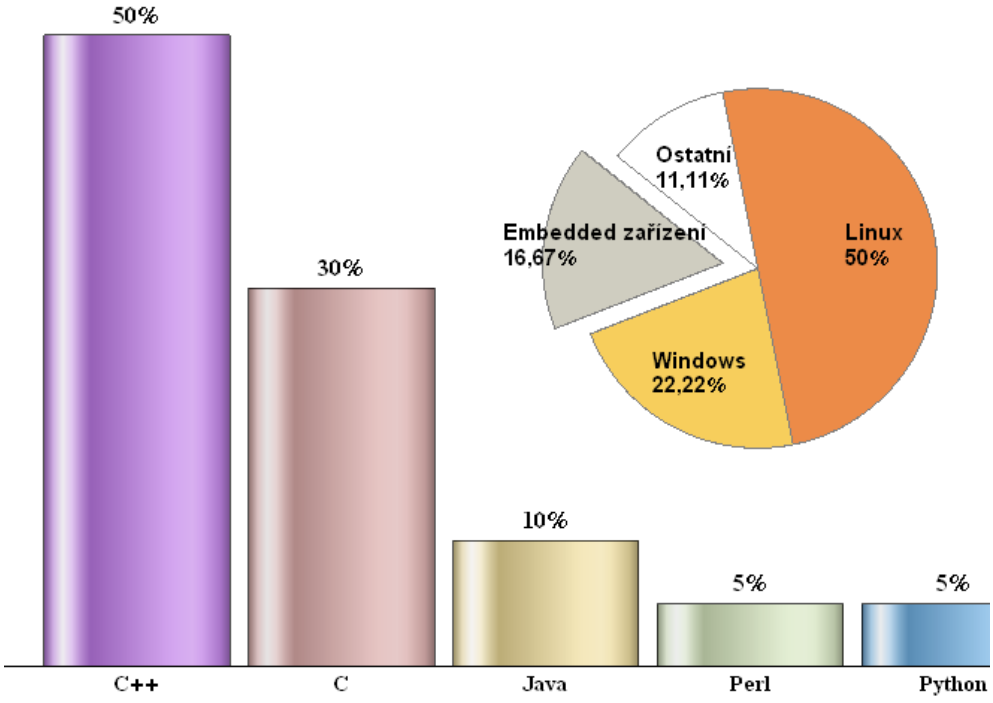


Obrázek 18 – Požadavky nadnárodních společností zaměřených na Linux

U nadnárodních společností, jejichž deklarovaná cílová platforma je založena na operačních systémech typu UNIX/Linux (viz Obrázek 18) převažuje poptávka po znalostech programovacího jazyka Java a programovacího jazyka C. Jazyk C je velmi blízký této platformě, protože v současnosti je jádro operačního systému Linux naprogramováno právě v tomto jazyce.



Obrázek 19 – Požadavky nadnárodních společností zaměřených na Windows



Obrázek 20 – Požadavky nadnárodních společností zaměřených na embedded zařízení

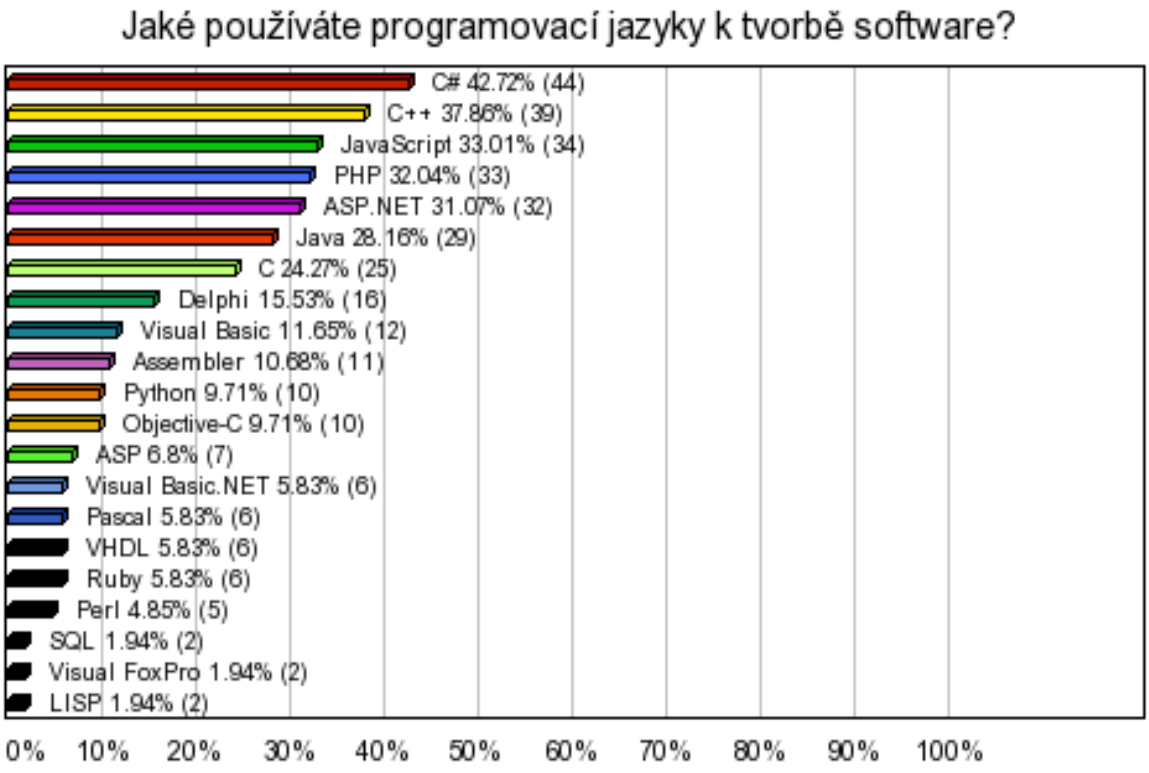
U nadnárodních společností, které se soustředí na platformu Windows (viz Obrázek 19) je stejně jako u českých společností v převaze jazyk C#. Je zde taky ve stejném poměru

programovací jazyk ASP.NET. Z výsledků vyplývá, že nadnárodní společnosti v oblasti cílové platformy Windows preferují platformu .NET.

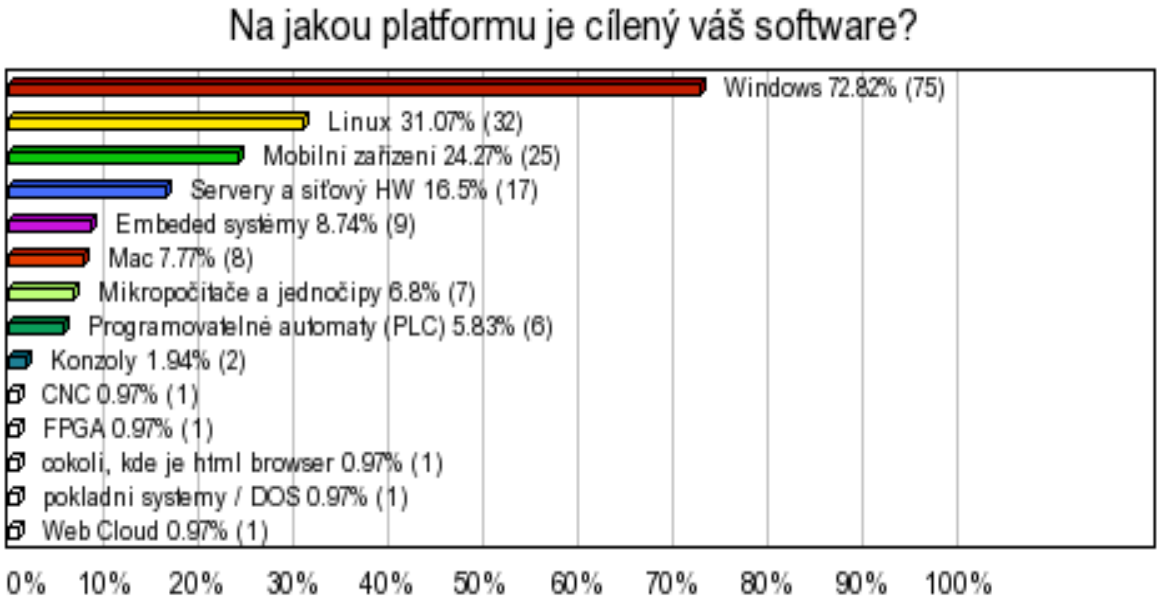
Další cílovou platformou jsou embedded zařízení (viz Obrázek 20), které jsou doménou programovacích jazyků C++ a C. U této platformy vestavěných zařízení se vyskytuje v hojné míře speciální hardware vyžadující přímé přístupy. Právě programovací jazyky založené na C/C++ jsou toho schopny.

4.4 Výsledky dotazníkového průzkumu

V tomto typu průzkumu byly samostatně osloveny softwarové a vývojové firmy, které se zabývají programováním vlastních aplikací. Bylo samostatně osloveno celkem 250 respondentů a vyhodnoceny byly celkem 103 záznamy. Veškeré výsledky jsou k dispozici v příloze na CD této diplomové práce. V následujících grafech jsou zachyceny podstatné výsledky z dotazníkového šetření empirického průzkumu.



Obrázek 21 – Výsledky dotazníkového šetření využití programovacích jazyků



Obrázek 22 – Výsledky dotazníkového šetření cílové platformy

Z výsledků dotazníkového šetření využití programovacích jazyků (viz Obrázek 21) vyplývá, že v průměru je v praxi nejvíce preferován programovací jazyk C#, tedy platforma založená na technologii .NET. Bylo potvrzeno preferování objektově orientovaných jazyků. V první šestici nejpoužívanějších programovacích jazyků je pouze jeden jazyk neinterpretovaný – jedná se o C++. Zajímavé je také hojné zastoupení jazyka C, který je považován za jazyk spíše nízkoúrovňový s podporou některých prvků abstrakce. Využití jazyka Java je oproti šetření požadavků trhu práce (viz Obrázek 11) aplikováno v menší míře.

Také cílové platformy (viz Obrázek 22), kterým je určen výsledný produkt se liší od požadavků trhu práce. Zatímco trh práce preferuje na 50% platformu Linux, oslovené firmy jsou zaměřeny přes 70% na platformu Windows společnosti Microsoft. Vzhledem k tomu, že v požadavcích trhu převažuje platforma Linux, dá se předpokládat, že v budoucnu bude tato platforma u softwarových firem nabírat na důležitosti na úkor ostatních platforem.

4.4.1 Dotazníkový průzkum v jednotlivých sektorech

Podobně jako v kapitole (Kapitola 4.3.2), byl i zde proveden dle zadání diplomové práce sektorový průzkum využití programovacích jazyků. Samotné výsledky jsou uloženy z prostorových důvodů v příloze P I.

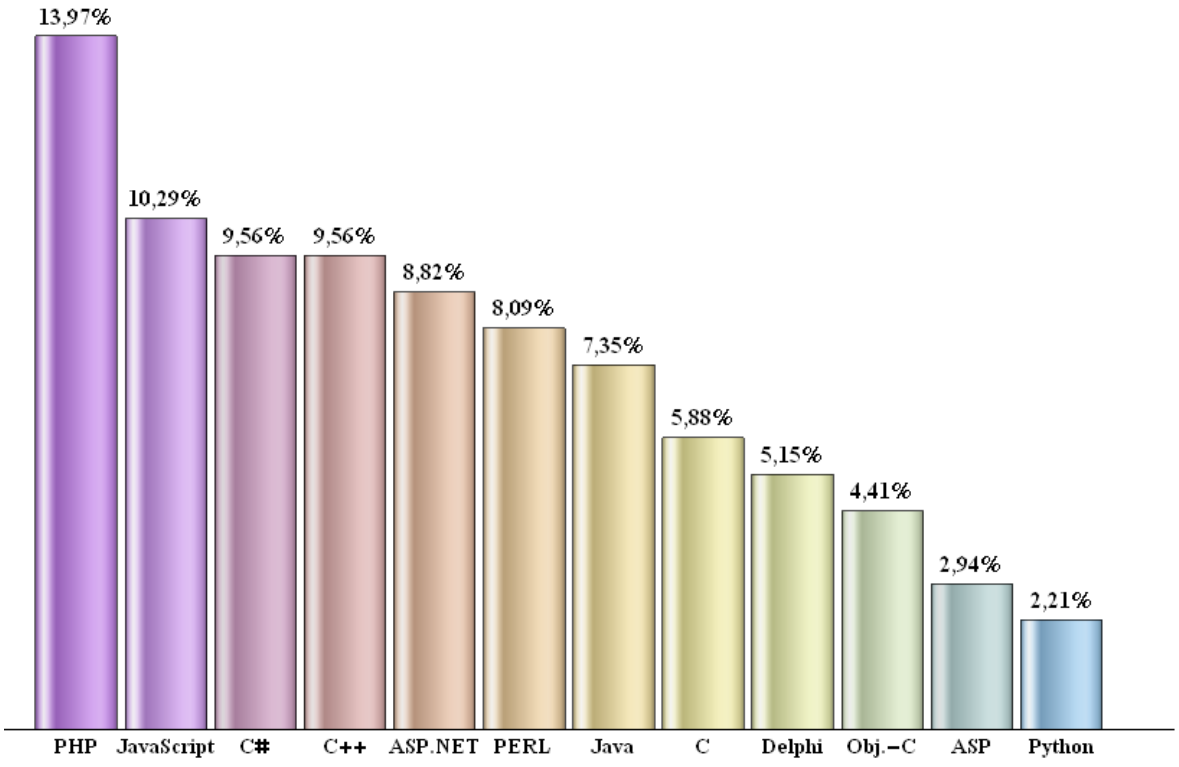
Z výsledků vyplývá, že současné používané programovací jazyky se liší od programovacích jazyků, které byly zjištěny průzkumem trhu. Současný pracovní trh preferuje ve většině případů znalosti programovacího jazyka Java. Avšak u zavedených firem není zatím využívána v takové míře. Lze tedy do budoucna předpokládat zvětšování podílu jazyka Java na úkor jiných programovacích jazyků. Z výsledků dotazníkového šetření totiž vyplývá, že přibližně jedna pětina firem se v budoucnu chystá změnit priority ve využívání programovacích jazyků, nebo o tom alespoň uvažují.

Naopak přibližně shodné výsledky jsou u programovacích jazyků na platformě .NET, především jazyka C#. Tento jazyk je hojně využíván jak v programátorské praxi oslovených firem a taky je jeden z nejžádanějších na trhu práce.

V obou případech je rovněž zřetelné, že je zde naprostá převaha objektově orientovaných jazyků.

4.4.2 Dotazníkový průzkum u českých společností

V následujících grafech jsou znázorněny výsledky průzkumu používání programovacích jazyků u ryze českých společností v rámci trhu.



Obrázek 23 – Ryze české společnosti s obratem menším než 10 milionů Kč

Windows	67,57 %
Linux	32,43 %
Mobilní zařízení	32,43 %
Serverová řešení	18,92 %
Mikropočítače a jednočipy	8,11 %
Programovatelné automaty	8,11 %
MAC OS	5,41 %
Emedded zařízení	5,41 %

Tabulka 16 – Cílové platformy českých společností s obratem menším než 10 milionů Kč

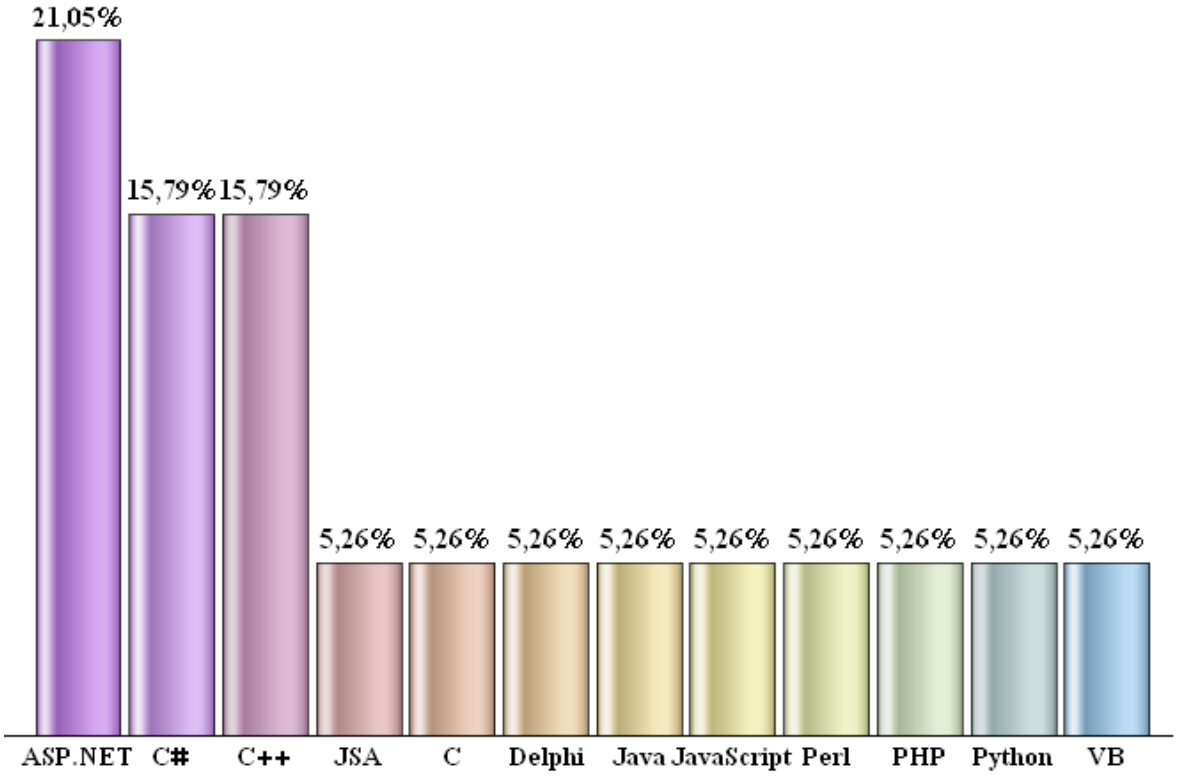
Z obrázku (Obrázek 23), kde je zaznamenáno prvních 12 nejpoužívanějších programovacích jazyků u menších českých softwarových firem z hlediska obratu, vyplývá, že v rámci jejich aktivit jsou nejvíce využívány programovací jazyky PHP a JavaScript, tedy především webová platforma. Jako další technologie, která má důvěru je .NET, který je zastoupen téměř z dvaceti procent. Z prvních šesti nejpoužívanějších jazyků je pouze jeden neinterpretovaný. Jedná se o C++.

Z tabulky (Tabulka 16) je zřejmé, že mezi cílovými platformami je opět nejvyužívanější platforma Windows. Téměř třetinové zastoupení mají produkty určené pro UNIX/Linux.

Rychlejší vývoj aplikace	80 %
Bohaté zkušenosti	80 %
Finance	80 %
Nenáročný na prostředky	60 %
Kvalitní frameworky	60 %
Dostatek programátorů	60 %
Dobrá budoucnost	40 %

Tabulka 17 – Důvody preference jazyka PHP u českých společností s obratem menším než 10 milionů Kč

Tabulka (Tabulka 17) obsahuje nejčastější důvody, které vedou ryze české společnosti s obratem pod 10 milionů k preferenci programovacího jazyka PHP.



Obrázek 24 – Ryze české společnosti s obratem větším než 10 milionů Kč a počtem zaměstnanců větším než 50

Windows	100 %
UNIX/Linux	22,22 %
Serverová řešení	22,22 %
Mobilní zařízení	11,11 %

Tabulka 18 – Cílové platformy velkých českých společností s obratem větším než 10 milionů Kč a počtem zaměstnanců větším než 50

Na obrázku (Obrázek 24) je zaznamenán výsledek dotazníkového šetření u velkých ryze českých společností. Je zde v převaze technologie interpretovaných jazyků .NET

a poměrně slušné zastoupení prezentuje i kompilovaný jazyk C++. Zbývající programovací jazyky mají rovnoměrné využití.

V tabulce (Tabulka 18) je procentuální zastoupení cílových platforem, pro které jsou určeny produkty těchto velkých českých společností. Bylo tedy zjištěno, že všechny velké české společnosti produkují software pro operační systém Windows. Pro systémy na bázi UNIX/Linux, či serverová řešení tvoří software přibližně jedna pětina z oslovených softwarových společností.

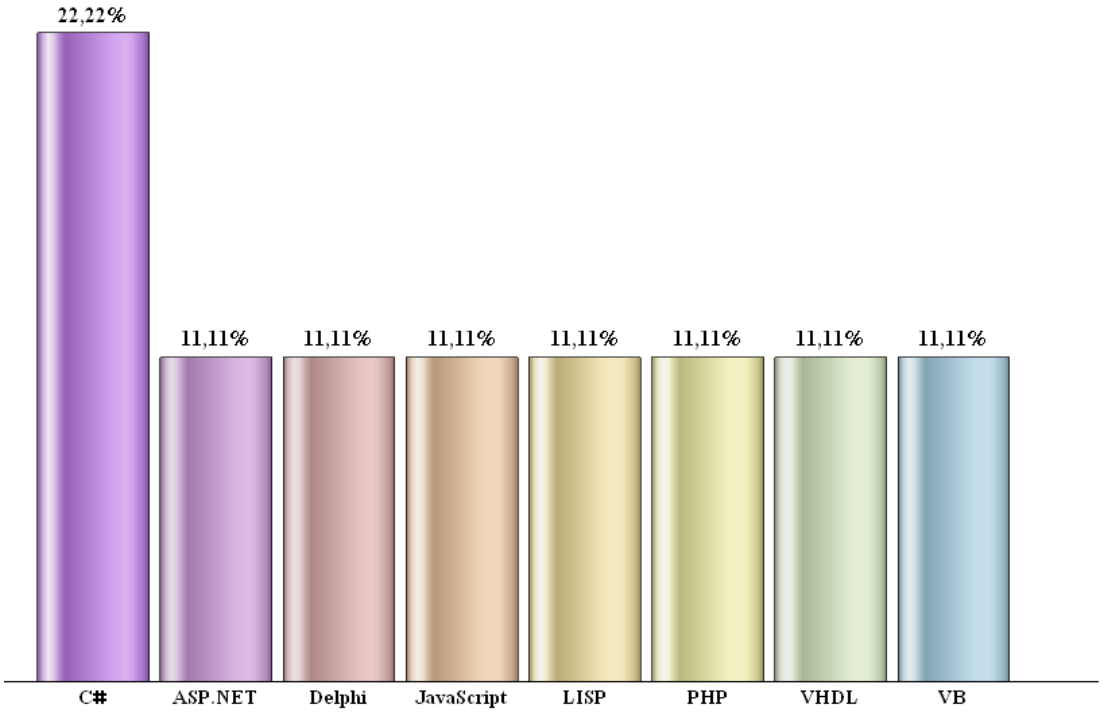
Rychlejší vývoj aplikace	75 %
Kvalitní knihovny	75 %
Spolehlivost, robustnost	50 %
Automatická správa prostředků	37,5 %
Dobrá budoucnost	37,5 %
Jednoduchost	25 %
Dostatek programátorů	25 %

Tabulka 19 – Důvody preference technologie .NET u českých firem s obratem nad 10 milionů Kč

V tabulce (Tabulka 19) jsou vyznačeny nejčastější důvody, které vedou ryze české firmy s obratem nad 10 milionů Kč k preferenci technologie .NET.

4.4.3 Dotazníkový průzkum u nadnárodních společností

V následujících grafech jsou znázorněny výsledky průzkumu používání programovacích jazyků u nadnárodních společností v rámci trhu.



Obrázek 25 – Nadnárodní společnosti s obratem menším než 10 milionů Kč

Windows	50 %
Mobilní zařízení	37,5 %
Emedded zařízení	25 %
Linux	12,5 %
Serverová řešení	12,5 %
Programovatelné automaty	12,5 %
Mikropočítače a jednočipy	12,5 %

Tabulka 20 – Cílové platformy nadnárodních společností s obratem menším než 10 milionů Kč

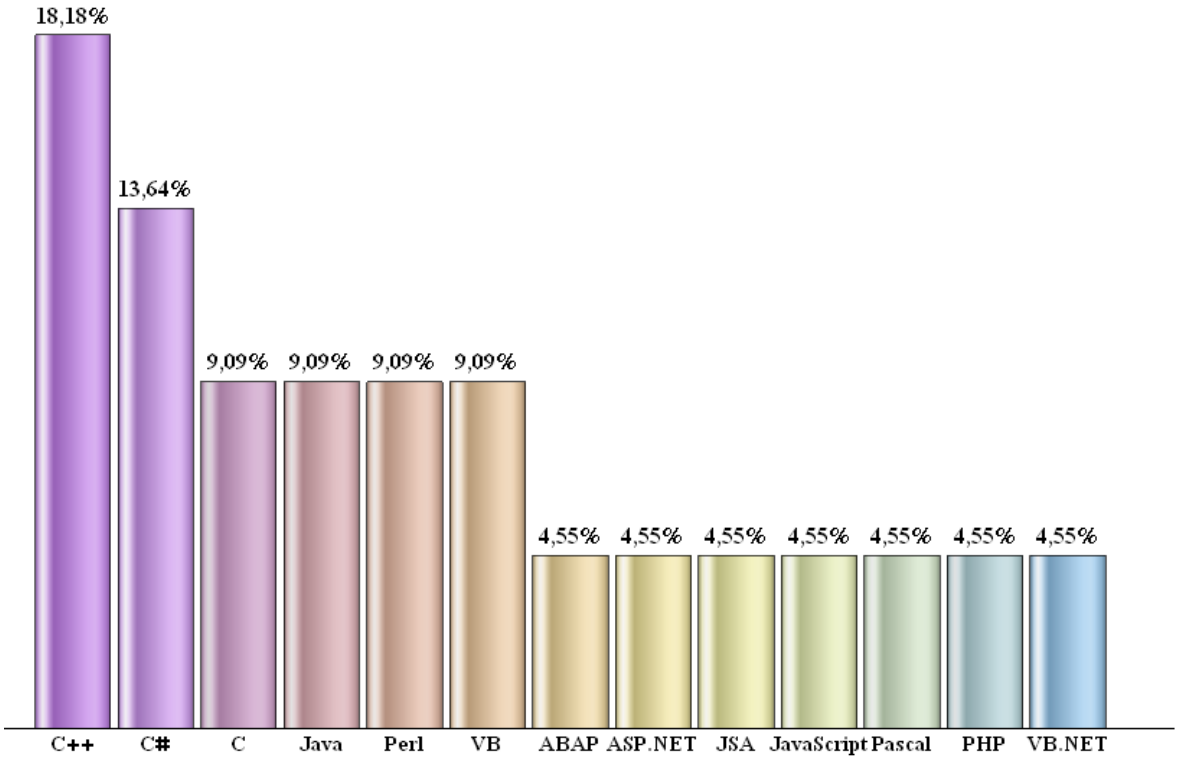
Na obrázku (Obrázek 25) je znázorněno využití programovacích jazyků u menších nadnárodních společností z hlediska obratu. I v tomto případě má podobně jako u ryze českých společností přednost platforma .NET. Zbývající jazyky jsou potom zastoupeny rovnoměrně v přibližném poměru 11 % na každý jazyk.

V tabulce (Tabulka 20) jsou znázorněny cílové platformy, pro které tyto společnosti produkují software. Opět ne v největší míře zastoupen operační systém firmy Microsoft. Mobilní platforma je v zastoupení větším než jedna třetina. Software pro zbývající platformy produkuje asi 12 % z těchto společností.

Rychlejší vývoj	50 %
Spolehlivost a robustnost	50 %
Nemožnost přejít na jiný jazyk	50 %
Užitečné knihovny	50 %

Tabulka 21 – Důvody preference technologie .NET u nadnárodních společností s obratem menším než 10 milionů Kč

Nadnárodní společnosti s obratem pod 10 milionů Kč, které preferují jazyk C# uvádějí své důvody podle tabulky (Tabulka 21).



Obrázek 26 – Nadnárodní společnosti s obratem větším než 10 milionů Kč a počtem zaměstnanců větším než 50

Windows	81,82 %
Linux	18,18 %
Mobilní zařízení	18,18 %
Programovatelné automaty	18,18 %
Serverová řešení	9 %

Tabulka 22 – Cílové platformy velkých nadnárodních společností s obratem větším než 10 milionů Kč a počtem zaměstnanců větším než 50

Z obrázku (Obrázek 26) je zřejmé, že velké nadnárodní společnosti preferují objektově orientovaný jazyk C++ společně s taky objektovým jazykem C#, který je ale interpretovaný a založený na platformě .NET, případně jeho open source variantě Mono.

Tabulka (Tabulka 22) vyjadřuje procentuální podíl cílových platforem, pro které jsou určeny produkty těchto nadnárodních firem. Stejně jako u ryze českých společností převažují produkty pro cílovou platformu Windows a podobně je taky zastoupena platforma UNIX/Linux, která společně s mobilními zařízeními a programovatelnými automaty tvoří asi jednu pětinu produkce.

Spolehlivost a robustnost	50 %
Bohaté zkušenosti	50 %
Nelze přejít na jiný jazyk	50 %

Tabulka 23 – Důvody preference jazyka C++ u nadnárodních společností s obratem větším než 10 milionů Kč

Tabulka (Tabulka 23) ukazuje nejčastější důvody, které vedou k preferování objektově orientovaného, neinterpretovaného jazyka C++ u velkých nadnárodních společností.

5 SHRNUTÍ VÝSLEDKŮ

V následující kapitole jsou celkově shrnuty výsledky, které byly získány v této diplomové práci a které naznačují směr, jakým se ubírají současné technologie na poli softwarového vývoje aplikací.

5.1 Užití programovacích jazyků

Při práci byly zjištěny současné a budoucí trendy ve využití programovacích jazyků. Zatímco ještě před několika lety byly v popředí neinterpretované programovací jazyky, především C++, v dnešní době se vývoj aplikací soustředí na jazyky interpretované. Výhoda je spatřována v mnohých dostupných knihovnách a rychlosti vývoje aplikace.

V budoucnosti budou žádány znalosti jazyků založených na .NET Frameworku, konkrétně se jedná o jazyk C# a ASP.NET. Cílové zaměření platforem je převážně směrováno na Windows.

V průzkumu trhu práce se jako nejžádanější vyskytoval jazyk Java. Z tohoto zjištění usuzuji, že v budoucích letech bude právě tento jazyk nabývat svůj podíl u současných vývojářských firem. Je to podloženo i dotazníkovým průzkumem, kdy jedna pětina firem uvažuje nebo se chystá na změnu priorit u aktuálně používaného jazyka.

U webových technologií budou i nadále v převaze jazyky PHP a JavaScript, jejichž oblíbenost je především u menších českých firem. Na trhu práce jsou v současnosti velmi žádané.

K výuce na vysokých školách tedy doporučuji výše zmíněné programovací jazyky s tím, že u jazyka Java je vhodné se zaměřit i na související technologie, které se k tomuto jazyku vyžadují (Obrázek 12).

Na dalším místě zájmu zůstane i jazyk C++, který ustupuje právě interpretovaným jazykům. V budoucnu ale bude stále žádaný, i když v menší míře než jazyky interpretované. Z toho důvodu by ve výuce programovacích jazyků neměl chybět ani jazyk C++.

5.2 Vývoj aplikace a jeho požadavky na profil programátora

V této kapitole diplomové práce jsou rozebrány aspekty vývoje aplikace jako celku a tomu odpovídající profil absolventa či uchazeče o post programátora. Jak již bylo zmíněno na několika místech této práce, k vývoji aplikace nejsou dostačující pouze znalosti programovacího jazyka, ale souvisí s tím i mnoho dalších věcí, bez kterých se programátor v dnešní době prostě neobejde.

Tyto aspekty jsou taky rozebírány v oboru zvaném softwarové inženýrství, které se právě věnuje komplexním postupům při vývoji software. Znalost softwarového inženýrství je předpoklad pro kvalitního vývojáře softwaru.

V této kapitole bude uvedeno několik postřehů k požadavkům na znalosti u současných vývojářů, které byly zjištěny v rámci sbírání údajů k dotazníkovému šetření. Existují samozřejmě různě drahé nástroje, které takto vedený vývoj podporují. Je ovšem třeba konstatovat, že nemůžou být využity, pokud zaměstnanci nejsou školou vycvičeni právě v určité samostatnosti a současně smyslu pro kolektivní práci (nejlépe v malých kroužcích, pro hlavní části vývoje ve dvojicích). Pokud neumí odhadnout, kdy do problému zapadli a je čas přizvat dalšího nebo nemají smysl pro určitou čistotu práce.

Jako ideální profil uchazeče o práci programátora lze kromě znalostí doporučených programovacích jazyků z kapitoly (Kapitola 5.1) definovat vlastnosti v následujících podkapitolách.

5.2.1 Získat od zákazníka smysluplné zadání

Získání takového zadání, které je proveditelné a které jasně definuje požadavky na výsledný produkt. Jedině správně získané zadání je možné použít v procesu vývoje software. V opačném případě dochází k finančním ztrátám.

5.2.2 Domluvit na dalším postupu bez právních aspektů

Dodržení licenčních podmínek vyvíjeného software. Taky je třeba dodržet dané podmínky nasazení software u cílového zákazníka.

5.2.3 Rychle proniknout do tématu, kterým se budu zabývat

Zde se jedná o problém jak klást otázky. Proniknutí do problému vede k jakési empatii se zákazníkem.

5.2.4 Připravit kostru projektu, odhadnout cykly vývoje, náklady

Navrhnutí kostry produktu úzce souvisí se správně definovaným zadáním (viz Kapitola 5.2.1). Kostra navržená na základě tohoto zadání slouží k odhadu jednotlivých cyklů vývoje, což vede k časovému odhadu do vydání prvních testovacích a konečných verzí produktu. Tato časová náročnost se odráží ve finančních nákladech, které budou vynaloženy při tvorbě zadaného projektu.

5.2.5 Odhadnout ideální nástroje pro vyhotovení

Zde se jedná mimo jiné právě o programovací jazyk. Určení použitých knihoven a produktů třetích stran. Definování vývojových nástrojů, které jsou ideální pro daný typ problému.

5.2.6 Uřídit vývoj projektu

Při samotném vývoji produktu je třeba sledovat jeho fáze a snažit se o dodržování daných standardů.

5.2.7 Týmová spolupráce

Samotný problém práce většího počtu lidí na projektu byl jedním z důvodů ke vzniku oboru zvaného softwarové inženýrství. Dobrá spolupráce v týmu, komunikace mezi vývojáři a správné rozdělení činností je nezbytné pro bezproblémový průběh vývoje aplikace.

5.2.8 Nástroje k testování kvality

Pro dokončený produkt je třeba definovat testy, které odhalí správnost výsledného produktu. Jedná se jak o správnost z hlediska zadání požadavků, tak o správnost chování směrem k uživateli.

5.2.9 Umět správně dekomponovat problém.

Jsou zde základní problémy při znalostech jednoduchých výpočtů v plovoucí čárce. Neznalost pojmů jako je odhad numerické stability, robustně pojaté programování a tak dále.

5.2.10 Znalosti návrhu databází pro IT

V případě, že aplikace bude spolupracovat s databází, je třeba umět navrhnout správnou strukturu a typ databáze. Bez tohoto nelze pokračovat ve vývoji, protože software je potom uzpůsobován databázovému schématu.

5.2.11 Schopnost napsat dokumentaci k softwarovému produktu

Dokumentace k softwarovému výrobku je nezbytná součást hotového produktu a správně napsaná a čtivá dokumentace je jednou z výhod softwarového produktu.

5.2.12 Více praktických úkolů ve výuce.

S tím, že někdo se ukáže jako lepší analytik, někdo jako matematik, jiný všechno domluví, aby se ostatní nehádali, další tomu dá vhodnou formu a podobně. Softwarové firmy nepotřebují, aby každý byl dobrý na všechno (což ani není v lidských silách), ale hlavně aby znal své meze.

ZÁVĚR

V průběhu práce byly postupně zjišťovány patřičné údaje potřebné k zhodnocení využití programovacích jazyků ve vývojové praxi. Byly prozkoumány jednotlivé sektory z pohledu odvětví a z těchto údajů byly vypočítány výsledky, které pomůžou při výuce programovacích jazyků na vysokých školách.

Ve zhodnocení výsledků práce bylo odpovězeno na otázky možné budoucnosti při vývoji aplikací a jmenovány konkrétní programovací jazyky a technologie, na které je třeba se v budoucnu zaměřit.

I když se v některých specializovaných sektorech odvětví stále využívá klasický přístup programování pomocí neinterpretovaných jazyků, je to spíše v menší míře, případně u přísně specializovaných úloh.

V dnešní době se v převaze projevily interpretované programovací jazyky, které v posledních letech i díky velkému rozvoji hardware zaznamenaly nárůst využití. Současné firemní prostředí vyžaduje, aby aplikace či produkt byl k dispozici co možná nejdříve a právě interpretované programovací jazyky díky kvalitním knihovnám nebo frameworku tuto možnost poskytují.

Vzhledem k tomu, že hotová komerční aplikace má životnost několik let, kdy je třeba tuto vylepšovat a provádět upgrade, znamená volba konkrétního programovacího jazyka rozhodnutí na dlouhou dobu dopředu. Díky této diplomové práci se může budoucí programátor nebo vysoká škola správně rozhodnout, kterým směrem ubírat výuku, aby bylo dosaženo maximálního uplatnění v praxi.

Že je v jednotlivých sektorech nespokojenost se současným stavem znalostí programovacích jazyků u nových uchazečů vyšlo i z dotazníkového průzkumu. Pouze 22 % respondentů odpovědělo, že mají dostatečné množství kvalitních programátorů.

V závěrečném zhodnocení diplomové práce byly tedy určeny konkrétní výsledky, konkrétní technologie. Je ovšem třeba zároveň s výukou programovacích jazyků zkoumat i související procedury, které jsou při vývoji aplikace nedílnou součástí znalostí programátora. Tyto znalosti jsou potřebné ve všech odvětvích a sektorech, pro které je daná aplikace vyvíjena. Jedná se především o schopnost práce v týmu a schopnost komunikace.

Je taky dobré umět odhadnout, jak dlouho bude trvat vývoj aplikace a umět navrhnout správnou strukturu budoucího produktu.

CONCLUSION

During the course of the thesis appropriate data necessary for evaluation of utilisation of programming languages in practical development is discovered. Individual sectors were explored from the viewpoint of branches and from this data results that will help in teaching of programming languages on universities and colleges were calculated.

Evaluation of results replies to the questions of possible future in application development and states specific programming languages and technologies that need to be concentrated upon in future.

Even though some specialised sectors of the field still use the classic approach to programming – i.e. using not interpreted languages - this is only in a smaller extent, mostly for highly specialised tasks.

Currently, interpreted programming languages prevail and thanks to rapid development of hardware they experience an increase in use. Contemporary company environments require applications and products to be available as soon as possible and it is interpreted programming languages that, thanks to quality libraries and framework, allow this possibility.

Taking into account that the life-time of a completed commercial application is several years, during which it needs to be upgraded, the choice of a programming language is a long time decision. By virtue of this diploma thesis a future programmer or a college or university can make a good decision in terms of what course they take in teaching so that maximum application in praxis is achieved.

The questionnaire query has shown that individual sectors are discontented with the current state of new job applicants' knowledge of programming languages. Only 22 % of respondents answered that they have sufficient amount of quality programmers.

The concluding part of the diploma thesis points out specific results - specific technologies. Along with teaching of programming languages, it is, however, necessary to explore the related procedures which form an essential knowledge of a programmer during the development of an application. This knowledge is needed in all branches and sectors for which the application is developed. This concerns mostly the ability to work in a team and

to communicate. The ability to estimate the time needed for the development of an application and to suggest correct structure of a future project is also useful.

SEZNAM POUŽITÉ LITERATURY

[1] LOUDEN, Kenneth, C. *Programming Languages : Principles and Practices*. 2nd Edition. [s.l.] : Thomson Brooks/Cole, 2003. 720 s. ISBN 0-534-95341-7.

[2] KADLEC, Josef. *Linuxsoft.cz* [online]. 16.7.2004 [cit. 2011-02-12]. Obecné pojednání o programovacích jazycích. Dostupné z WWW: <http://www.linuxsoft.cz/article.php?id_article=268>.

[3] LIBERTY, Jesse. *Naučte se C++ za 21 dní*. Praha : Computer Press, 2002. 766 s. ISBN 80-7226-774-4.

[4] *Wikidot.com* [online]. c2011 [cit. 2011-02-13]. Programovací paradigmata. Dostupné z WWW: <http://ari.wikidot.com/programovaci-paradigmata>.

[5] *Státnice na FM TUL* [online]. 24. 11. 2010 [cit. 2011-02-13]. Funkcionální programování a Lambda kalkulus. Dostupné z WWW: http://statnice.obrys.cz/index.php?title=Funkcion%C3%A1ln%C3%AD_programov%C3%A1n%C3%AD_a_Lambda_kalkulus.

[6] *Tiobe Software* [online]. February 2011 [cit. 2011-02-12]. TIOBE Programming Community Index. Dostupné z WWW: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>.

[7] FALTÝNEK, Lukáš. *LinuxEXPRES* [online]. c 2007 [cit. 2011-03-13]. Programovací jazyky nejen v Linuxu. Dostupné z WWW: <http://www.linuxexpres.cz/praxe/programovaci-jazyky>.

[8] KAČMÁŘ, Dalibor. *Jazyk C : učebnice pro střední a vysoké školy*. Praha : Computer Press, 2001. 186 s. ISBN 80-7226-295-5.

[9] HEROUT, Pavel. *Učebnice jazyka C. IV.přepracované vydání*. České Budějovice : Kopp, 2005. 271 s. ISBN 80-7232-220-6.

[10] PRATA, Stephen. *Mistrovství v C++. 2. aktualizované vydání*. Brno : Computer Press, 2004. 1006 s. ISBN 80-251-0098-7.

[11] HARMS, Daryl; MCDONALD, Kenneth. *Začínáme programovat v jazyce Python. 2.opravené vydání*. Brno : Computer Press, 2008. 449 s. ISBN 978-80-251-2161-0.

[12] PONKRÁC, Miloslav. *PHP a MySQL : bez předchozích znalostí*. Brno : Computer Press, 2007. 221 s. ISBN 978-80-251-1758-3.

[13] VIRIUS, Miroslav. *C# pro zelenáče*. Praha : Neocortex, 2002. 255 s. ISBN 80-86330-11-7.

[14] MABBUTT, Dan. *About.com* [online]. c2011 [cit. 2011-02-16]. About Visual Basic with Definitions, History, and VB Background. Dostupné z WWW: <http://visualbasic.about.com/od/vbnetspecialtopics/a/aboutvb.htm>.

[15] *Programming for iOS* [online]. 09.03.2010 [cit. 2011-02-15]. Objective-C: A Brief History. Dostupné z WWW: <http://cupsofcocoa.wordpress.com/2010/09/03/objective-c-a-brief-history/>.

[16] JANOVSKÝ, Dušan. *Jakpsatweb.cz* [online]. 14. února 2011 [cit. 2011-02-15]. Úvod do JavaScriptu. Dostupné z WWW: <http://www.jakpsatweb.cz/javascript/javascript-uvod.html>.

[17] KYSELA, Martin. *Perl : Kompletní kapesní průvodce programátora*. Praha : Grada Publishing, 2005. 134 s. ISBN 80-247-1170-2.

[18] *Ruby community*. Ruby Programming Language [online]. c2010 [cit. 2011-02-15]. Dostupné z WWW: <http://www.ruby-lang.org/en/>.

[19] YOUNG, Stephen J. *Programovací jazyky pro RT-aplikace*. 1. vid. Praha : SNTL, 1988. 338 s.

[20] PUNCH, Keith, F. *Základy kvantitativního šetření*. Vyd. 1. Praha : Portál, 2008. 150 s. ISBN 978-80-7367-381-9.

[21] FORET, Miroslav. *Marketingový průzkum : Poznáváme svoje zákazníky*. Vydání první. Brno : Computer Press, 2008. 120 s. ISBN 978-80-251-2183-2.

[22] *Nakladatelství Portál* [online]. c2011 [cit. 2011-03-27]. Metody sběru dat. Dostupné z WWW: <http://www.portal.cz/scripts/detail.php?id=24159>.

[23] *Dotaznik-online.cz* [online]. c 2007 [cit. 2011-03-13]. Dostupné z WWW: <http://www.dotaznik-online.cz/>.

[24] *Dotaznik-online.cz* [online]. c 2007 [cit. 2011-03-13]. Dotazník - Struktura dotazníku. Dostupné z WWW: <http://www.dotaznik-online.cz/zaklady-dotazniku.htm>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ActiveX	Rámec pro definici znovupoužitelných softwarových komponent v programovacím jazyce nezávislým způsobem
ANSI	American National Standards Institute – Americký národní standardizační institut
ByteCode	Je mezijazyk vytvořený pro potřeby Java Virtual Machine, do kterého se překládají zdrojové kódy pro běh na platformě Java Virtual Machine
CAD	Computer Aided Design – počítačové navrhování a projektování
CAM	Computer Aided Manufacturing – počítačová podpora výroby
CGI	Common Gateway Interface – protokol pro propojení externích aplikací s webovým serverem
ECMA	European Computer Manufacturers Association – Evropské sdružení výrobců počítačů
Embedded	Vestavěné zařízení
Framework	Softwarová struktura pro podporu programování, vývoje aplikací či spouštění aplikací
GIS	Geografický informační systém
HTML	HyperText Markup Language – Hypertextový značkovací jazyk
HTTP	HyperText Transfer Protocol – Síťový protokol určený pro výměnu a sdílení informací
ISO	International Organization for Standardization – Mezinárodní organizace pro normy
IT	Informační technologie
JIT	Just In Time Compiler – Způsob práce s mezikódem na cílové platformě. Spočívá v tom, že tzv. mezikód (též bytecode) může být přeložen v době instalace, v době spuštění nebo jsou překládány jenom části v okamžiku, kdy jsou potřeba.

JVM	Java Virtual Machine – Modul virtuálního stroje využívaný ke spuštění dalších počítačových programů a skriptů vytvořených v jazyce Java. Úkolem tohoto modulu je zpracovat pouze tzv. mezikód, který je v Javě označován jako Java bytecode
Open source	Software, jehož zdrojový kód je volně přístupný a za dodržení jistých licenčních podmínek je možné ho upravit podle potřeby
XML	Extensible Markup Language – rozšiřitelný značkovací jazyk

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

CD Obsahuje původní materiály a grafy získané dotazníkovým šetřením a průzkumem trhu, dále je zde originál diplomové práce

P I Dotazníkový sektorový průzkum využití programovacích jazyků

PŘÍLOHA P I: DOTAZNÍKOVÝ SEKTOROVÝ PRŮZKUM

V této příloze jsou zaznamenány priority ve využití programovacích jazyků u softwarových firem v rámci ČR. Jedná se o výsledky v jednotlivých sektorech.

Automatizace a řízení			
C++	12,79 %	VHDL	2,33 %
PHP	10,47 %	ABAP	1,16 %
C#	9,30 %	ASP	1,16 %
Perl	9,30 %	COBOL	1,16 %
ASP.NET	8,14 %	GX Developer	1,16 %
C	6,98 %	Impulse-C	1,16 %
JavaScript	6,98 %	Objective-C	1,16 %
JSA (Assembler)	4,65 %	Pascal	1,16 %
Delphi	4,65 %	Python	1,16 %
Visual Basic	4,65 %	Ruby	1,16 %
Java	3,49 %	STEP 7	1,16 %
Visual Basic.NET	3,49 %	Verilog	1,16 %

Tabulka 1 – Dotazníkový průzkum v sektoru automatizace a řízení

Automobilní průmysl			
C#	12,82 %	Visual Basic	2,56 %
ASP.NET	10,26 %	ABAP	2,56 %
JavaScript	10,26 %	ASP	2,56 %
JSA (Assembler)	7,69 %	CAPL	2,56 %
C	7,69 %	Visual FoxPro	2,56 %
C++	7,69 %	Impulse-C	2,56 %
Perl	7,69 %	PHP	2,56 %
VHDL	7,69 %	Verilog	2,56 %
Java	5,13 %		

Tabulka 2 – Dotazníkový průzkum v sektoru automobilního průmyslu

Bankovnictví			
ASP.NET	12,73 %	Visual Basic	5,45 %
C#	12,73 %	C	3,64 %
JavaScript	12,73 %	Python	3,64 %
Java	9,09 %	Cobol	1,82 %
PHP	9,09 %	LISP	1,82 %
ASP	5,45 %	Pascal	1,82 %
C++	5,45 %	VHDL	1,82 %
Delphi	5,45 %	Visual Basic.NET	1,82 %
Perl	5,45 %		

Tabulka 3 – Dotazníkový průzkum v sektoru bankovnictví

Doprava			
C#	15,38 %	Visual Basic	3,85 %
ASP.NET	9,62 %	ASP	1,92 %
C++	9,62 %	Delphi	1,92 %
JavaScript	9,62 %	Impulse-C	1,92 %
Java	7,69 %	Objective-C	1,92 %
Perl	7,69 %	Uniface	1,92 %
PHP	7,69 %	Verilog	1,92 %
C	5,77 %	VHDL	1,92 %
JSA (Assembler)	3,85 %	Visual Basic.NET	1,92 %
Ruby	3,85 %		

Tabulka 4 – Dotazníkový průzkum v sektoru dopravy

Ekonomika a obchod			
C#	14,42 %	Pascal	1,92 %
ASP.NET	11,54 %	Ruby	1,92 %
JavaScript	11,54 %	Visual Basic	1,92 %
C++	10,58 %	ASP	0,96 %
Java	10,58 %	JSA (Assembler)	0,96 %
PHP	10,58 %	Bash script	0,96 %
Perl	5,77 %	FAND	0,96 %
C	4,81 %	Uniface	0,96 %
Python	3,85 %	Visual Basic.NET	0,96 %
Delphi	1,92 %	DataFlex	0,96 %
Objective-C	1,92 %		

Tabulka 5 – Dotazníkový průzkum v sektoru ekonomiky a obchodu

Elektrotechnický průmysl			
C#	13,79 %	ASP.NET	3,45 %
C++	13,79 %	Delphi	3,45 %
C	10,34 %	Impulse-C	3,45 %
Perl	10,34 %	Java	3,45 %
JSA (Assembler)	6,90 %	PHP	3,45 %
JavaScript	6,90 %	Verilog	3,45 %
VHDL	6,90 %	Visual Basic	3,45 %
ABAP	3,45 %	Visual Basic.NET	3,45 %

Tabulka 6 – Dotazníkový průzkum v sektoru elektrotechnického průmyslu

Energetika			
C#	17,86 %	ABAP	
C++	14,29 %	JSA (Assembler)	3,57 %
JavaScript	10,71 %	Delphi	3,57 %
ASP.NET	7,14 %	PHP	3,57 %
C	7,14 %	Visual Basic	3,57 %
Java	7,14 %	Visual Basic.NET	3,57 %
Perl	3,57 %		

Tabulka 7 – Dotazníkový průzkum v sektoru energetiky

GIS a geografické systémy			
C#	15,38 %	LISP	3,85 %
JavaScript	15,38 %	Objective-C	3,85 %
ASP.NET	11,54 %	Perl	3,85 %
C++	11,54 %	Python	3,85 %
JSA (Assembler)	7,69 %	Ruby	3,85 %
Java	7,69 %	VHDL	3,85 %
PHP	7,69 %		

Tabulka 8 – Dotazníkový průzkum v sektoru GIS

Herní software			
C#	11,43 %	C	5,71 %
JavaScript	11,43 %	Delphi	5,71 %
Objective-C	11,43 %	ASP	2,86 %
Perl	11,43 %	ASP.NET	2,86 %
PHP	11,43 %	JSA (Assembler)	2,86 %
C++	8,57 %	Python	2,86 %
Java	8,57 %	Ruby	2,86 %

Tabulka 9 – Dotazníkový průzkum v sektoru herního softwaru

Letecký průmysl			
ASP.NET	16 %	PHP	8 %
C#	16 %	Ruby	8 %
JavaScript	12 %	Delphi	4 %
C	8 %	Java	4 %
C++	8 %	Pascal	4 %
Perl	8 %	Visual Basic.NET	4 %

Tabulka 10 – Dotazníkový průzkum v sektoru leteckého průmyslu

Oblast informačních technologií			
C++	11,93 %	Ruby	2,75 %
JavaScript	11,01 %	Visual Basic.NET	2,75 %
PHP	11,01 %	ABAP	0,92 %
ASP.NET	9,17 %	ASP	0,92 %
C#	8,26 %	JSA (Assembler)	0,92 %
Java	8,26 %	Asterisk CMD	0,92 %
Perl	7,34 %	Bash script	0,92 %
Python	5,50 %	COBOL	0,92 %
C	4,59 %	LISP	0,92 %
Visual Basic	3,67 %	Pascal	0,92 %
Delphi	2,75 %	VHDL	0,92 %
Objective-C	2,75 %		

Tabulka 11 – Dotazníkový průzkum v sektoru software pro oblast IT

Ochrana a bezpečnost			
Java	21,05 %	C#	5,26 %
C++	15,79 %	Delphi	5,26 %
ASP.NET	10,53 %	Perl	5,26 %
JavaScript	10,53 %	PHP	5,26 %
JSA (Assembler)	5,26 %	Python	5,26 %
C	5,26 %	VHDL	5,26 %

Tabulka 12 – Dotazníkový průzkum v sektoru bezpečnostního software

Státní správa			
JavaScript	14,52 %	ASP	3,23 %
Java	11,29 %	Delphi	3,23 %
C#	9,68 %	Python	3,23 %
PHP	9,68 %	Visual Basic.NET	3,23 %
ASP.NET	8,06 %	ABAP	1,61 %
Visual Basic	8,06 %	Bash script	1,61 %
C++	6,45 %	FAND	1,61 %
Perl	6,45 %	Pascal	1,61 %
C	4,84 %	Ruby	1,61 %

Tabulka 13 – Dotazníkový průzkum v sektoru státní správy

Stavebnictví			
C#	20 %	Delphi	10 %
C++	20 %	Java	10 %
JavaScript	20 %	Visual Basic	10 %
ASP.NET	10 %		

Tabulka 14 – Dotazníkový průzkum v sektoru stavebnictví

Strojírenství			
C#	14,29 %	ASP.NET	7,14 %
C++	14,29 %	ABAP	3,57 %
VHDL	14,29 %	Java	3,57 %
JSA (Assembler)	10,71 %	JavaScript	3,57 %
C	10,71 %	LISP	3,57 %
Perl	10,71 %	PM	3,57 %

Tabulka 15 – Dotazníkový průzkum v sektoru strojírenství

Software na zakázku			
C#	10,61 %	Visual Basic	2,23 %
JavaScript	10,61 %	Visual Basic.NET	2,23 %
PHP	10,61 %	Assembler	1,68 %
ASP.NET	10,06 %	Pascal	1,68 %
C++	9,50 %	Bash script	1,12 %
Java	7,82 %	Visual FoxPro	1,12 %
Perl	7,26 %	CAPL	0,56 %
C	5,59 %	COBOL	0,56 %
Delphi	3,35 %	FAND	0,56 %
Python	3,35 %	GX Developer	0,56 %
Objective-C	2,79 %	STEP 7	0,56 %
ASP	2,23 %	VHDL	0,56 %
Ruby	2,23 %	DataFlex	0,56 %

Tabulka 16 – Dotazníkový průzkum v sektoru software na zakázku

Software pro specializovaná hardware (ovladače)			
C++	33,33 %	Perl	16,67 %
Assembler	16,67 %	PHP	16,67 %
C	16,67 %		

Tabulka 17 – Dotazníkový průzkum v sektoru specializovaného SW pro daný HW

Školství			
ASP.NET	11,43 %	Visual Basic.NET	8,57 %
C#	11,43 %	C	5,71 %
JavaScript	11,43 %	Perl	5,71 %
C++	8,57 %	LISP	2,86 %
Delphi	8,57 %	Pascal	2,86 %
Java	8,57 %	Visual Basic	2,86 %
PHP	8,57 %	DataFlex	2,86 %

Tabulka 18 – Dotazníkový průzkum v sektoru školství

Telekomunikace			
C	11,11 %	JavaScript	9,52 %
C++	11,11 %	ASP.NET	6,35 %
Objective-C	11,11 %	JSA (Assembler)	3,17 %
Perl	11,11 %	Asterisk CMD	1,59 %
PHP	11,11 %	Delphi	1,59 %
C#	9,52 %	Python	1,59 %
Java	9,52 %	VHDL	1,59 %

Tabulka 19 – Dotazníkový průzkum v sektoru telekomunikací

Zdravotnictví			
ASP.NET	10,34 %	C	5,17 %
C#	10,34 %	Delphi	5,17 %
JavaScript	10,34 %	Pascal	5,17 %
PHP	10,34 %	Perl	5,17 %
Java	8,62 %	ASP	3,45 %
C++	6,90 %	JSA (Assembler)	1,72 %
Visual Basic	6,90 %	Visual FoxPro	1,72 %
Visual Basic.NET	6,90 %	DataFlex	1,72 %

Tabulka 20 – Dotazníkový průzkum v sektoru zdravotnictví